Date of acceptance Grade

Instructor

# Structural EM methods in phylogenetics and stemmatology

Yuan Zou

Helsinki November 16, 2010

M.Sc. Thesis

UNIVERSITY OF HELSINKI
Department of Computer Science

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

| Tiedekunta — Fakultet — Faculty | Laitos — Institution — Department |
|---|---|
| Faculty of Science | Department of Computer Science |

Tekijä — Författare — Author
Yuan Zou

Työn nimi — Arbetets titel — Title
Structural EM methods in phylogenetics and stemmatology

Oppiaine — Läroämne — Subject
Computer Science

| Työn laji — Arbetets art — Level | Aika — Datum — Month and year | Sivumäärä — Sidoantal — Number of pages |
|---|---|---|
| M.Sc. Thesis | November 16, 2010 | 66 pages + 9 appendices |


Tiivistelmä — Referat — Abstract

The purpose of the thesis is to discuss the structural EM (SEM) method in stemmatic and phylogenetic analyses. The structural EM method was introduced and developed by Friedman since 1998 [17]. It reconstructs maximum likelihood networks by integrating structure search in the traditional EM method for maximum likelihood analysis. Generally, it can be applied for learning all kinds of Bayesian networks. The phylogenetic and stemmatic trees are special cases of Bayesian networks. The Bayesian networks are decomposable to local elements. To increase the speed of searching a reasonable structure, the maximum spanning tree algorithm can be incorporated in the structural EM method. Furthermore, the simulated annealing method is applied to alleviate the problem of local optima.

Phylogenetic trees are assumed to be binary trees and the observed data are located only in leaf nodes. On the other hand, stemmatic trees are not confined to be binary trees, and possibly with known documents in the internal nodes. Other computational methods, which are designed for phylogenetic analysis, are usually confined to reconstruct binary trees. However, the structural EM method is suitable for stemmatic tree reconstructions because it can find arbitrary tree structures.

In the experiments of analyzing the artificially generated phylogenetic data sets, the structural EM method obtained comparable results to other computational methods. In the experiment of clustering the regulatory genes in *streptococcus pneumoniae*, it also generated biological consistent results. For stemmatic analysis, the structural EM method performed consistently well for all artificially data sets. Furthermore, the method was very suitable for analyzing the large data set, the *Heinrichi* data set, in stemmatology. However, some other computational methods, for example, the neighbor-joining method, the least-squares method, and the maximum a posteriori (MAP) method using simulation algorithms performed unsatisfactory for the larger data set. In the future, we need to further explore several aspects of the structural EM method such as how to learn the edge lengths in stemmatic analysis, and how to identify the root of a tree.

ACM Computing Classification System (CCS):
G.3 [PROBABILITY AND STATISTICS ],
H.1.1 [Systems and Information Theory]

Avainsanat — Nyckelord — Keywords
The structural EM method, Stemmatology, Phylogenetics, Bayesian networks

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — övriga uppgifter — Additional information

# Contents

# 1 Introduction

The evolutionary tree reconstruction is one of the central tasks in studies of phylogenetics and stemmatology. In phylogenetics, a set of DNA or protein sequences are analyzed to infer the evolutionary relationships of species. Similarly in stemmatology, the main aim is to recover the relationships of the sources and copies from a collection of incomplete textual documents. Usually, before the computational programs can analyze the sequences or documents, they need to be aligned to a matrix with each column containing a set of homologous elements and each row containing one sequence or document.

In the past years, many computational methods have been developed to find the true tree structure, like the neighbor-joining method [54], the least-squares method [13], the parsimony method [12], the maximum likelihood (ML) method [9], the maximum a posteriori (MAP) method [50], and the RHM method [51]. One of the well established methods in tree reconstruction is the maximum likelihood (ML) method, which tries to find a tree structure with the maximum posterior likelihood. There are several types of algorithms for the maximum likelihood methods. One is the structural EM (SEM) method introduced and developed by Friedman since 1998 [17]. It reconstructs the maximum likelihood tree by integrating structure search in the traditional EM method for maximum likelihood analysis. In the experiments, the structural EM method performed well in both phylogenetic and stemmatic analyses.

The thesis first reviews the procedures of the structural EM method. Then, in the experiments in phylogenetic and stemmatic analyses, we compare the performances of the structural EM method with other computational methods. Finally, we discuss several aspects that need to be developed for the structural EM method in the future. In the appendix, we focus on the preprocessing step of how to get a reliable multiple sequence alignment in phylogenetic and stemmatic analyses.

# 2 Bayesian networks and Structural EM

Phylogenetic trees and stemmatic trees are variants of Bayesian networks. Generally, the structural EM method can be applied for learning all kinds of Bayesian networks. In this chapter, we first briefly introduce the basics of Bayesian networks. Then, we discuss the structural EM method for learning general Bayesian networks and its applications for phylogenetic and stemmatic inferences.

## 2.1 Bayesian networks

A Bayesian network is a directed acyclic graph that models probabilistic relationships among a set of variables. This section first introduces general properties and then focuses on the Markov property [19] of Bayesian networks.

A Bayesian network encodes joint probability distributions of a set of variables in their domain [26]. A node in the graph model represents a variable, and a directed edge indicates that a variable immediately affects its descendant. On a domain $\mathbf{X} = \{x_1, ..., x_n\}$, a Bayesian network represents two sets of relationships, including the conditional independences between variables and the local probability distributions of variables. The Markov property of Bayesian networks is that, given a node's parents, it is independent of its non-descendants in the network. Thus, denoting the parents of a node $x_i$ as $\Pi_i$, the probability of a Bayesian network with n nodes can be factorized by the chain rule as

$$P(\mathbf{X}) = \prod_{i=1}^{n} P(x_i|\Pi_i). \tag{1}$$

Figure 1 shows a simple Bayesian network about the relationships between the raining (R), the observation of wet grass (G), and the using of sprinkler (S). From the figure, we can derive the joint distribution of the three variables as

$$P(G, S, R) = P(G|S, R)P(S|R)P(R). \tag{2}$$

From Equation (2) and by Bayes' theorem, we can perform various inferences, for example, what is the probability of raining if the grass is wet.

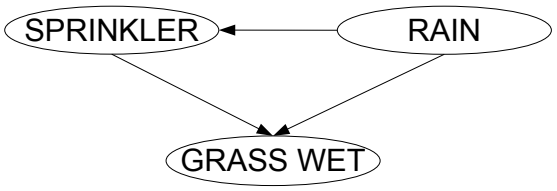



Figure 1: An example of Bayesian networks.

Bayesian networks can be applied for encoding expert systems or causal relationships. Compared to other expert systems such as the decision tree and the rule

based system, Bayesian networks represent the missing data statistically and can provide higher prediction accuracies [26]. This is crucial when the missing data is correlated with the observed data. Most models for encoding expert systems fail to generate unbiased predictions because their inabilities to encode the dependencies between correlated variables. Bayesian networks can also handle hidden variables which interact with observed variables [26]. Furthermore, Bayesian networks inferred by Bayesian methods or other models like MDL [61] can avoid over the fitting of the data. Nowadays, Bayesian networks are widely used in both supervised and unsupervised learnings. They are applied in various fields such as computational biology, document classification, image processing, information retrieval, gaming, law, *etc.*.

## 2.2 Learning Bayesian networks

This section focuses on a Bayesian approach for inferring the structure and local probabilities in Bayesian networks. It includes learning from the complete data and the incomplete data.

### 2.2.1 Complete data

Given a data set $D = \{\mathbf{X}^1, ..., \mathbf{X}^N\}$ containing $N$ observations, the task of learning the Bayesian network is to find the network with structure and parameters that best match the data. Generally, the score function is introduced and the learning strategy is designed to find the network model that optimizes the score. One criterion of the score function, the belief scoring function, is the marginal likelihood (MLL) function of $P(D|M)$ [21]. Denoting a model as $M \in \mathbf{M}$ in the model space $\mathbf{M}$, the model parameters as $\Theta_M = \{\theta_M^1, ..., \theta_M^k\}$, and $\hat{\Theta}_M$ as the parameters which best fit the data under model $M$, the score function can be written as

$$MLL(M, \hat{\Theta}_M : D) = \log P(D|M, \hat{\Theta}_M) = \log \frac{P(\hat{\Theta}_M, D|M)}{P(\hat{\Theta}_M|D, M)}. \tag{3}$$

The second equation is derived by Bayes Theorem. Another score function is the BIC/MDL scoring function. The belief scoring function approximates it when the data set is large enough to be treated as a multivariate-Gaussian distribution [56]. The BIC/MDL score is

$$BIC(M, \hat{\Theta}_M : D) = \log P(D|\hat{\Theta}_M, M) - \frac{d}{2} \log N, \tag{4}$$

where $d$ is the number of the parameters of the model $M$. The first term in the score function is the log-likelihood of data $D$ given the model $M$ and its best fit parameters. The second term is the penalization term for preferring simpler models. This criterion avoids the over-fitting problem caused by choosing more complex models with larger number of parameters.

When the network structure is known, there is a closed-form solution for the parameters which optimize the score function. The penalization term in Equation (4) is fixed because the network structure is defined. To calculate the first term, for so called decomposable models like Bayesian networks, Equation (3) can be written as [17]

$$MLL(M, \hat{\Theta}_M : D) = \sum_{j=1}^{N} \sum_{x_i^j, \Pi_i^j} N(x_i^j, \Pi_i^j) \log(\hat{\theta}_{x_i^j, \Pi_i^j}) \ , \tag{5}$$

where $N(x_i^j, \Pi_{x_i^j})$ is the number of the occurrences of the pair of $\{x_i^j, \Pi_i^j\}$ in data $D$. Because the score function is a linear function of each $\log \hat{\theta}_{x_i^j, \Pi_i^j}$, the best parameters can be solved locally by

$$\hat{\theta}_{x_i^j, \Pi_i^j} = \frac{N(x_i^j, \Pi_i^j)}{N(\Pi_i^j)}. \tag{6}$$

This indicate that to find the best structure, we can use local search strategies. For example, the local search can be performed by doing a random edge transformation each time with the constraint that no directed cycle is created [55]. The transformation includes adding, removing and reversing of edges. In the first two cases, only the local score of the node that the edge points to needs to be updated. In the last case, the scores of the nodes linked by the edge need to be re-evaluated. The modification is accepted if the network score is increased each time until no improvement can be made. This hill-climbing search strategy is efficient for search through the model space but can be stuck at local optima. To address this problem, other methods such as simulated annealing (SA) or Gibbs' sampling can be applied to help escape the local optima [27].

### 2.2.2 Incomplete data

There are three mechanisms which underline the missing data: Missing Completely at Random (MCAR), Missing at Random (MAR) based on the observed data, and Not Missing at Random (NMAR) [25]. For the last situation, a model for the missing data needs to be constructed before processing the data. For the first two cases, the simplest way is the missing data imputation. However, the simple data imputation

can not represent the uncertainty of the missing data and leads to serious bias to the analysis. More sophisticated statistical methods can provide better solutions to the missing data problems.

Considering the case of MAR in learning Bayesian networks, the linearity of the score function is no longer valid because different observations in the data set are no longer independent. Thus, the solutions of the best parameters for a given model are no longer closed forms. It is much harder to predict the best model. To solve the score function, several approximation methods can be made. For example, one method is the stochastic sampling method like the Markov chain Monte Carlo (MCMC) method [20]. But usually, this method is time consuming and inefficient because it needs to generate a large sample set before convergence. Another approximation is based on the Laplace's approximation of the sampled data. It assumes that the posterior of the parameter based on the large sample data is peaked, thus it can be approximated as a Gaussian density. But it is not suitable for the sparse data and can cause strong biases. In this thesis, we discuss an efficient method in the framework of the Bayesian approach, the structural EM method for learning Bayesian networks and handling the missing data. The structural EM method optimizes the score directly by a hill-climbing search strategy. To alleviate the problem of local optima in this method, the simulated annealing method is incorporated in the learning process.

## 2.3 Structural EM

In this section, we first prove the convergence of the structural EM method theoretically. Then, we discuss how the structural EM method is applied in learning Bayesian networks.

The structural EM method can be applied in both the discrete data and the continuous data which is smoothly distributed. The observed data is denoted as $\mathbf{O}$, and the hidden data is represented as $\mathbf{H}$. The learning task of the structural EM is to select the model that has the maximum Bayesian score given the data. Using the belief score function and denote

$$Q(M, \Theta : M_l, \Theta_l) = E[\log P(\mathbf{H},\mathbf{O}|M, \Theta)|M_l, \Theta_l], \tag{7}$$

as the expected Bayesian score of model $M$ based on the parameters inferred in the previous step $l$. Assuming that we can estimate the complete data likelihood given a model, a general outline of the structural EM method is given below [17]:

1: Initialize $M_0$ and $\Theta_0$ randomly.
2: **repeat**
3: For each $M$, compute the expected Bayesian score $Q(M, \Theta : M_l, \Theta_l)$ based on the model and parameters inferred in the previous step. {E-step}
4: Select a new model $M_{l+1}$ and parameters $\Theta_{l+1}$ that maximize the expected Bayesian score by

$$\{\Theta_{l+1}, M_{l+1}\} = \arg\max_{\Theta, M} Q(M_{l+1}, \Theta : M_l, \Theta_l). \tag{8}$$

{M-step}
5: **until** The expected Bayesian score converged:

$$Q(M, \Theta : M_l, \Theta_l) = Q(M, \Theta : M_{l+1}, \Theta_{l+1}), \tag{9}$$

or run out of time.

In the structural EM method, the real score can be optimized by iteratively increasing the expected Bayesian score. The convergence of the algorithm can be proved by the following statement by Friedman [17]

$$\log P(\mathbf{O}|M_{l+1}, \Theta_{l+1}) - \log P(\mathbf{O}|M_l, \Theta_l) \geq Q(M_{l+1}, \Theta_{l+1} : M_l, \Theta_l) - Q(M_l, \Theta_l : M_l, \Theta_l). \tag{10}$$

From this statement, we can see that the increase of the logarithm of the Bayesian score is always equal or larger than the increase of the expected Bayesian score. Thus, we can improve the real Bayesian score by improving the expected Bayesian score. The proof of this statement is an extension of the proof of the convergence of the parametric EM algorithm [41]. The proof is based on the following steps:

$$\begin{aligned}
&\log \frac{P(\mathbf{O}|M_{l+1}, \Theta_{l+1})}{P(\mathbf{O}|M_l, \Theta_l)} \\
&= \log \sum_{\mathbf{H}} \frac{P(\mathbf{H}, \mathbf{O}|M_{l+1}, \Theta_{l+1})}{P(\mathbf{O}|M_l, \Theta_l)} \cdot \frac{P(\mathbf{H}|\mathbf{O}, M_l, \Theta_l)}{P(\mathbf{H}|\mathbf{O}, M_l, \Theta_l)} \\
&= \log \sum_{\mathbf{H}} P(\mathbf{H}|\mathbf{O}, M_l, \Theta_l) \cdot \frac{P(\mathbf{H}, \mathbf{O}|M_{l+1}, \Theta_{l+1})}{P(\mathbf{H}, \mathbf{O}|M_l, \Theta_l)} \\
(*) &\geq \sum_{\mathbf{H}} P(\mathbf{H}|\mathbf{O}, M_l, \Theta_l) \cdot \log \frac{P(\mathbf{H}, \mathbf{O}|M_{l+1}, \Theta_{l+1})}{P(\mathbf{H}, \mathbf{O}|M_l, \Theta_l)} \\
&= E[\log \frac{P(\mathbf{H}, \mathbf{O}|M_{l+1}, \Theta_{l+1})}{P(\mathbf{H}, \mathbf{O}|M_l, \Theta_l)} | M_l, \Theta_l] \\
&= Q(M_{l+1}, \Theta_{l+1} : M_l, \Theta_l) - Q(M_l, \Theta_l : M_l, \Theta_l)
\end{aligned} \tag{11}$$

.

The inequality by (*) is derived by Jensen's inequality. Based on this statement, when it is hard to find the best model which maximizes the expected Bayesian score, we can further modify the M-step to a more relaxed selection criterion:

$$Q(M_{l+1}, \Theta_{l+1} : M_l, \Theta_l) > Q(M_l, \Theta_{l+1} : M_l, \Theta_l). \tag{12}$$

In this manner, the structural EM method can always find a model that is not worse than the previous one. However, the structural EM method does not guarantee to find the model with the best score globally. One reason of being stuck in the sub-optimal model is that the sub-optimal model can generate parameters with distributions that always make itself better scored than other models. Therefore, methods such as the simulated annealing (SA) method which can help the structural EM method to escape local optima are necessary.

To further derive the standard structural EM applied in Bayesian networks, recall that we need to calculate the expected Bayesian score by evaluating the marginal probability: $\log P(\mathbf{H}, \mathbf{O}|M, \Theta)$ as though we have complete data. In learning Bayesian networks, when we have complete data, we can decompose the score locally as a function of the probabilities of the set of factors $F = \{f_1, ...f_m\}$ of model $M$

$$E[\log P(\mathbf{H}, \mathbf{O}|M, \Theta)] = \sum_{k=1}^{m} E[\log f_k] \approx \sum_{k=1}^{m} \log E[f_k]. \tag{13}$$

The second approximation is exact when $\log f_k$ has linear arguments. Thus, the Bayesian expected score can be refined as

$$Q(M, \Theta : M_l, \Theta_l) = \sum_{k=1}^{m} \log E[f_k|M_l, \Theta_l]. \tag{14}$$

Generally, the MAP parameters can be computed by the parametric EM method or gradient descent algorithms. For updating the tree structure, as presented in section 2.2.1, we can use the local search method which only changes a small portion of tree structure each time. In the section of the structural EM method for learning phylogenetic and stemmatic trees, we discuss a more efficient method for updating tree structure with a detailed illustration.

# 3 Structural EM in phylogenetics

In this chapter, we discuss the structural EM method in phylogenetics. First, we introduce the background knowledge of phylogenetics and mathematical models applied in phylogenetics. Then, we review how the structural EM method is applied

in phylogenetics. Finally, we present the experiments of the structural EM method in phylogenetic field.

## 3.1 Introduction to Phylogenetics

In phylogenetics, the aim is to reconstruct a species tree that reflects the evolutionary process of a group of species. A species tree can be inferred from one or several gene trees. The gene tree, which is studied in molecular phylogenetics, is the tree structure that is learned upon the evolutionary relationships of genes. To simplify the issue of reconstructing a species tree, a gene tree can be treated as a species tree under several presumptions. First, tree branches are long enough so that the within group differences can be ignored. Second, gene copies coalesce within species before speciation events. Last, no events of lateral transfers, gene combinations, and random extinctions occur [7]. In this situation, the gene tree has the same topology as the species tree [35], and only one individual is needed as the sample for inferring the phylogenetic tree.

The phylogenetic analysis of reconstructing a gene tree can be generalized in 5 steps.

- First, choose suitable molecular markers. Different markers can lead to major differences in the resulting trees. The choice of molecular markers depends on the study purpose and the sequence property. For example, to study closely related organisms, DNA sequences, which evolve much faster than protein sequences, can be used. However, in the case of study of widely divergent groups, protein sequences or ribosomal RNA sequences are preferred.
- Perform the multiple sequence alignment, which aims to make aligned elements genealogically related.
- Choose a mathematical model for the molecular evolution.
- Construct the phylogenetic tree.
- Perform statistical analyses for assessing the reliability of the resulting tree. First, the reliability of the tree can be accessed by resampling methods such as bootstrapping. Second, to find whether one tree is significantly better than another, statistical tests, such as the Kishino-Hasegawa test [33] or the Shimodaira-Hasegawa test [57], can be performed.

Among the 5 steps, the first and last step are out of the range of this thesis. They are not explained further. The multiple sequence alignment is discussed in the appendix.

## 3.2 Mathematical models in phylogenetics

In this section, we review mathematical models for molecular evolution and how the models are incorporated in the structural EM method.

### 3.2.1 CTMC for molecular evolution

In the structural EM method for phylogenetic analysis, we need to know $p_{a\to b}(t)$, which indicates the probability of $a$ evolving to $b$ in the time $t$. Different mathematical models are defined for calculating $p_{a\to b}(t)$. In phylogenetics, the evolution process is assumed as a continuous-time Markov chain (CTMC) [40]. It assumes that the probability of mutating to a molecule only depends on the previous state of that molecule. The Markov property of the sequence evolution can be written as

$$p(x(t_n) = b|x(t_1) = a_1, ..., x(t_{n-1}) = a_{n-1}) = p(x(t_n) = b|x(t_{n-1}) = a_{n-1}). \quad (15)$$

The CTMC is further assumed to be time-homogeneous with the following property

$$p(x(t-s) = b|x(0) = a) = p(x(t) = b|x(s) = a). \quad (16)$$

Through simple inference, this function results in the Chapman-Kolmogorov equation [46], which is a very important property needed in the structural EM method for phylogenetic analysis. Represent the space of elements as $\Sigma$, $a$, $b$, and $c$ are elements in $\Sigma$, we get

$$p(x(t+s) = b|x(0) = a) = \sum_{c\in\Sigma} p(x(t+s) = b|x(s) = c)p(x(s) = c|x(0) = a). \quad (17)$$

The CTMC model has another important property, the reversibility, which is also needed in the structural EM method. Denote the state as $\pi$, the reversibility can be written as

$$\pi_a p_{a\to b}(t) = \pi_b p_{b\to a}(t). \quad (18)$$

### 3.2.2 Transition probabilities

In this section, we discuss how to calculate the transition probabilities of elements in phylogenetics. The contents are mainly based on [40]. The Chapman-Kolmogorov

equation indicates that a matrix $P(t)$ containing transition probabilities between all states is a semigroup of stochastic matrices. This semigroup is assumed to be a standard semigroup which has the property

$$p_{a\to b}(t) > 0. \tag{19}$$

For a standard semigroup, denoteing the *instantaneous transition rate* or the *transition intensity* of CTMC as $q_{ab}$, $p_{a\to b}(t)$ can be represented as

$$\begin{cases} p_{a\to b}(t) = q_{ab}(t) + o(t), \\ p_{a\to a}(t) = 1 + q_{aa}(t) + o(t). \end{cases}, \tag{20}$$

where o(t) (small ordo) is a function such that $o(t)/t \to 0$ when $t \to 0$. From the definition, it holds:

$$\begin{cases} \sum\limits_{b\in\Sigma} q_{ab} = 0, \\ \sum\limits_{b\neq a: b\in\Sigma} q_{ab} = q_{aa}. \end{cases}. \tag{21}$$

A matrix $\varrho$ with $q_{ab}$ as elements is called the *generator* of CTMC, which is usually used for representing different mathematical models for molecular evolution. For example, the *generator* of the generic model for the DNA evolution is

$$\varrho = \begin{pmatrix} -q_{11} & q_{12} & q_{13} & q_{14} \\ q_{21} & -q_{22} & q_{23} & q_{24} \\ q_{31} & q_{32} & -q_{33} & q_{34} \\ q_{41} & q_{42} & q_{43} & -q_{44} \end{pmatrix}. \tag{22}$$

Combining Formula (20) and (21) we have

$$\frac{p_{a\to b}(t+s) - p_{a\to b}(t)}{s} = \sum_{c\in\Sigma} p_{a\to c}(t) q_{cb} + o(s)/s, \tag{23}$$

when $t \to 0$, $\frac{p_{a\to b}(t+s)-p_{a\to b}(t)}{s}$ approximates the first derivative with respect to $t$:

$$p'_{a\to b}(t) = \sum_{c\in\Sigma} p_{a\to c}(t) q_{cb}. \tag{24}$$

Representing Formula (24) in matrix form, we have

$$P'(t) = P(t)\varrho. \tag{25}$$

Resolve this equation, then

$$P(t) = e^{t\varrho}, \tag{26}$$

where

$$e^{t\varrho} = \sum_{l=0}^{\infty} \frac{t^l}{l!} \varrho^l. \tag{27}$$

It is obvious that we can put a scaling value $\mu$ in Equation (26), and get

$$P(t) = e^{(\mu t)(\frac{1}{\mu}\varrho)}. \tag{28}$$

Hence in the CTMC model, no absolute time can be used to define the mutation matrix. But the scaling can be done to get the average intensities of mutations and the one step mutation probability matrix. We can define a finite number $\lambda$, such that

$$q_{ab} \leq \lambda < \infty, \text{ for all a, b.} \tag{29}$$

Define the one step mutation probability matrix as $\overline{P}$,

$$\overline{P} = I + \frac{1}{\lambda}\varrho. \tag{30}$$

Let $N(t)$ be a Poisson process with the density $\lambda$, then the general transition probability matrix $P(t)$ can be represented as

$$P(t) = \sum_{l=0}^{\infty} \overline{P}^l Pois(N(t) = l), \tag{31}$$

where the right hand is a continuous time random process that runs the Poisson process with the density $\lambda$ in a continuous time period. The one-step transition matrix moves from one state to another when the Poisson process jumps with one step [40].

### 3.2.3 Substitution models

As most amino acid substitution models have parallels in nucleotide substitution models, this section focuses only on nucleotide substitution models. The General Time-Reversible (GTR) model [63] is introduced first, because most nucleotide substitution models can be derived by imposing restrictions on the parameters of GTR model. After that, the Felsenstein F81 model (F81) [9], the Kimura Two-Parameter model (K2P) [32], and the Jukes-Cantor model (JC) [44] are introduced.

The *generator* of GTR model can be defined as

$$\varrho = \begin{pmatrix} * & \mu\mathbf{a}\pi_2 & \mu\mathbf{b}\pi_3 & \mu\mathbf{c}\pi_4 \\ \mu\mathbf{a}\pi_1 & * & \mu\mathbf{d}\pi_3 & \mu\mathbf{e}\pi_4 \\ \mu\mathbf{b}\pi_1 & \mu\mathbf{d}\pi_2 & * & \mu\mathbf{f}\pi_4 \\ \mu\mathbf{c}\pi_1 & \mu\mathbf{e}\pi_2 & \mu\mathbf{f}\pi_3 & * \end{pmatrix}. \tag{32}$$

Here the diagonal elements are represented as * for simplicity. They can be calculated by the second equation of (21). The parameter $\mu$ is the average instantaneous substitution rate, which is the same for all possible substitutions. Parameter $\mu$ is usually set to be 1. Parameters **a, b, c, d, e**, and **f** tunes $\mu$ to assign different rates for substitutions between different base pairs. Parameter $\pi$ is the equilibrium distribution of each base. Each column of $\varrho$ shares the same $\pi$, which indicates the rate of changing to one base is proportional to the equilibrium probability of that base. Also, it is clear that the GTR model is reversible. The GTR model is on the top of a hierarchically nested model set, with other models obtained by assigning constraints to its parameters.

When setting $\mathbf{a} = \mathbf{b} = \mathbf{c} = \mathbf{d} = \mathbf{e} = \mathbf{f} = 1$, we get the F81 model

$$\varrho = \begin{pmatrix} * & \pi_2 & \pi_3 & \pi_4 \\ \pi_1 & * & \pi_3 & \pi_4 \\ \pi_1 & \pi_2 & * & \pi_4 \\ \pi_1 & \pi_2 & \pi_3 & * \end{pmatrix}. \tag{33}$$

The F81 model allows four nucleotides to have unequal frequencies at equilibrium, but it neglects the rate variations between different bases. Contrary to the F81 model, the K2P model takes rate differences into account by assuming that all bases have the same probability at equilibrium. In molecular biology, the change between a purine and a pyrimidine, namely the transversion, is usually more severe and less common than the transition, which is the substitution occurring within purines or pyrimidines. In the K2P model, with the considering of the rate difference between transition and transversion, we set

$$\begin{cases} \mathbf{a} = \mathbf{b} = \mathbf{e} = \mathbf{f} = 1 \\ \mathbf{c} = \mathbf{d} = \dfrac{\alpha}{\beta} \\ \pi_1 = \pi_2 = \pi_3 = \pi_4 \end{cases}, \tag{34}$$

where $\alpha$ is the transition intensity and $\beta$ is the transversion intensity. Usually we have $\alpha > \beta$. Thus, the *generator* of K2P model is

$$\varrho = \begin{pmatrix} * & \beta & \beta & \alpha \\ \beta & * & \alpha & \beta \\ \beta & \alpha & * & \beta \\ \alpha & \beta & \beta & * \end{pmatrix}. \tag{35}$$

The JC model is the simplest model with no arbitrary equilibrium probabilities permitted and no rate variations considered. It can be obtained by setting $\mathbf{a} = \mathbf{b} =$

$\mathbf{c} = \mathbf{d} = \mathbf{e} = \mathbf{f} = 1$ and $\pi_1 = \pi_2 = \pi_3 = \pi_4$. The *generator* of the JC model is

$$\varrho = \begin{pmatrix} * & \alpha & \alpha & \alpha \\ \alpha & * & \alpha & \alpha \\ \alpha & \alpha & * & \alpha \\ \alpha & \alpha & \alpha & * \end{pmatrix}. \tag{36}$$

In this thesis, the three models mentioned above are utilized by the structural EM method for the phylogenetic tree reconstruction.

### 3.2.4 Phylogenetic tree as a special Bayesian network

The phylogenetic tree is a special Bayesian network. Each node in the phylogenetic tree denotes one species. The topology of the tree represents the evolutionary process. The edge length between a pair of nodes indicates the period between the separation of the linked species. In molecular phylogenetics, the species tree is usually inferred by reconstructing the evolutionary tree of gene sequences. In the phylogenetic tree, a species is conditionally independent of its non-descendants given its ancestor species. For instance, conditioned on the character state of a species' ancestor, the character in this species evolves independent of others species.

In the phylogenetic analysis, the separation time determines the probability of mutation from one molecule to another when the evolution model is set. Thus, the parameters $\Theta$ in the structural EM method for phylogenetics is the edge lengths $\mathbf{t}$. We then denote the edge length between nodes $i$ and $j$ as $t_{i,j}$.

In the phylogenetic analysis, usually a set of sequences are first aligned to a matrix with each row containing one sequence as a variable, and each column or site as one case of the observed variable. In addition, each site is usually assumed to be independent along the sequence. Refer the appendix for the methods for multiple alignment of phylogenetic data. Represent a data set containing $N$ DNA or protein sequences as $S = \{S_1, ..., S_N\}$. One sequence $S_i$ containing $M$ amino acids or nucleotides can be denoted as $S_i = \{x_i^1, ..., x_i^M\}$, where $x_i^j$ is the molecule in the site $j$ in the sequence $S_i$.

In the phylogenetic analysis, known sequences are assumed to be located in leaf nodes and the task is to infer a binary tree with internal nodes which contains missing data for sequences of the ancestors. Given $N$ leaf nodes, there are $N-2$ internal nodes, that can be denoted as $\{x_{N+1}^j, ..., x_{2N-2}^j\}$ at site $j$, in a binary tree. Therefore, reconstructing a phylogenetic tree can be viewed as find the best restricted structure

from an incomplete sequence data set. Before we discuss how to derive the Bayesian score function for the phylogenetic tree, we need some properties derived from the models of phylogenetic evolution.

In molecular phylogenetics, the mutations of different elements, for example, the amino acids in protein sequences and nucleotides in DNA sequences, are assumed to follow the continuous-time Markov chain (CTMC) model (Section 3.2.1). There are various mathematical models representing different assumptions in evolution processes under this general model. The special cases are discussed in section 3.2. Here are some general properties which are needed for inferring the Bayesian score function:

- Denote the space of molecule states as $\Sigma$, and $\{a, b, c\}$ as the states in this space. The Chapman-Kolmogorov equation [46] holds for the evolution of molecules:

$$p_{a\to b}(t+t') = \sum_{c\in\Sigma} p_{a\to c}(t)p_{c\to b}(t'). \tag{37}$$

  This equation indicates that there is no memory in the evolution process. If the initial and end states are fixed, the probability of the evolution is the same no matter which internal path is chosen. It can also be presented in a matrix form. Denote a transition matrix $P$ as the mutation probabilities of different states, the Chapman-Kolmogorov equation can also be written as

$$P(t+t') = P(t)P(t') = P(1)^{t+t'}, \tag{38}$$

  where $P(1)$ is the matrix of one step of the changing of molecules. Thus, the probability matrix of the evolution over a duration $t$ is the one step changing matrix to the power of $t$.

- The evolution process is reversible for the initial and end states,

$$p_a p_{a\to b}(t) = p_b p_{b\to a}(t). \tag{39}$$

  In terms of the equilibrium of the evolution, it implies that the probability of the first sequence evolving to the second one and the probability of the reverse process are the same. Thus, without extra information, we can not determine the time order of two sequences only by their compositions under the general model for molecular evolution.

Based on the general evolution model, we can then infer the conditional probability of the phylogenetic tree. Since the sites evolve independently, we first consider only

one site in the sequence matrix. Suppose that there are $2N-1$ random variables $\{x_1, ..., x_{2N-1}\}$ where the first $N$ variables are the observed sequences, and the last $N-2$ variables are hidden variables representing the internal divergence processes. The hypothetical topology for a phylogenetic tree is represented by an undirected tree $\mathbf{T}$, which is described by its set of edges. $(i,j) \in \mathbf{T}$ indicates that $\mathbf{T}$ has an edge between $i$ and $j$. Then, we randomly pick up one node as root $r$. All the edges are in the direction away from the root. Denoting the parent of $i$ as $\Pi_i$, we can calculate the probability of the whole tree by factorizing it on each node:

$$P(x_1, ..., x_{2N-1}|\mathbf{T}, \mathbf{t}) = P(x_r) \prod_{i \neq r} P(x_i|x_{\Pi_i}, t_{i,\Pi i}). \tag{40}$$

By the reversibility of the evolution process, we have for any pair of connected nodes

$$\frac{P(x_i|x_j, t_{i,j})}{P(x_i)} = \frac{P(x_j|x_i, t_{j,i})}{P(x_j)}. \tag{41}$$

As a result, the direction of the tree can be ignored and the probability of whole tree does not depend on which root we pick at the beginning

$$P(x_1, ..., x_{2N-1}|\mathbf{T}, \mathbf{t}) = \left[\prod_i P(x_i)\right] \left[\prod_{(i,j)\in\mathbf{T}} \frac{P(x_i|x_j, t_{i,j})}{P(x_i)}\right]. \tag{42}$$

By marginalizing the whole tree probability on internal nodes, the probability of observed data in one single site given the structure can be calculated as

$$P(x_1, ..., x_N|\mathbf{T}, \mathbf{t}) = \sum_{x_{N+1}} ... \sum_{x_{2N-2}} P(x_1, ..., x_{2N-1}|\mathbf{T}, \mathbf{t}). \tag{43}$$

After calculating the probability over one site, we can compute the conditional probability of the whole observed sequence data $\mathbf{O}$ by multiplying the marginal probabilities over all $m$ sites. Denoting $x_i^j$ as node $i$ in site $j$, we have

$$P(\mathbf{O}|\mathbf{T}, \mathbf{t}) = \prod_{k=1}^{m} P(x_1^k, ..., x_N^k|\mathbf{T}, \mathbf{t}). \tag{44}$$

## 3.3 Structural EM for phylogenetic tree reconstruction

In this section, we first discuss how to derive the expected Bayesian score function for phylogenetic tree reconstruction. After that, we review several aspects for efficiently applying the structural EM method in the phylogenetic field in practice. This section is mainly based on the work of Friedmam who introduced the structural EM method

and applied it in SEMPHY program for phylogenetic tree reconstruction [17, 18]. The structural EM method for phylogenetics can be summarized as the architecture in Figure 2.

### 3.3.1 Expected Bayesian score

In the structural EM method, the expected Bayesian score is used to optimize the real Bayesian score calculated from the incomplete data, that usually cannot be optimized directly. Recall that in the phylogenetic analysis, known sequences are assumed to be located in leaf nodes. Assume there is a complete data set including the observed sequences ($\mathbf{O}$) in leaf nodes and the hidden sequences ($\mathbf{H}$) in internal nodes. Denote all possible states as $\Sigma$. Then, the logarithm of the marginal likelihood function is

$$\begin{aligned} L(\mathbf{O},\mathbf{H}|\mathbf{T},\mathbf{t}) &= \sum_{k=1}^{m} \log P(x_1^k,...,x_{2N-2}^k|\mathbf{T},\mathbf{t}) \\ &= \sum_{k=1}^{m}\left[\sum_{i=1}^{2N-2}\log P(x_i^k) + \sum_{(i,j)\in\mathbf{T}}\log\frac{P(x_i^k|x_j^k,t_{i,j})}{P(x_i^k)}\right] \\ &= \sum_{i=1}^{2N-2}\sum_{a\in\Sigma}\sum_{k=1|x_i^k=a}^{m}\log P_a + \sum_{(i,j)\in\mathbf{T}}\sum_{(a,b)\in\Sigma^2}\sum_{k=1|(x_i^k,x_j^k)=(a,b)}^{m}\log\frac{P_{a\to b}(t_{i,j})}{P_b}\ . \end{aligned} \tag{45}$$

The term $\sum_{k=1|x_i^k=a}^{m}\sum_{i=1}^{2N-2}\log P_a$ is a constant. It can be denoted as $C$. The second term can be written as the count of co-occurrences of two molecules, which is the normalized conditional probability. Denote $N$ as the count, therefore

$$L(\mathbf{O},\mathbf{H}|\mathbf{T},\mathbf{t}) = C + \sum_{(i,j)\in\mathbf{T}}\sum_{(a,b)\in\Sigma^2} N(x_i=a,x_j=b)[\log P_{a\to b}(t_{i,j}) - \log P_b]. \tag{46}$$

Thus, the likelihood function can be optimized locally on each edge. Denoting the likelihood of an edge pair as a weight $w_{i,j}(t_{i,j})$, then the likelihood function can be further simplified as

$$L(\mathbf{O},\mathbf{H}|\mathbf{T},\mathbf{t}) = C + \sum_{(i,j)\in\mathbf{T}} w_{i,j}(t_{i,j}), \tag{47}$$

where

$$w_{i,j}(t_{i,j}) = \sum_{(a,b)\in\Sigma^2} N(x_i=a,x_j=b)[\log P_{a\to b}(t_{i,j}) - \log P_b]. \tag{48}$$

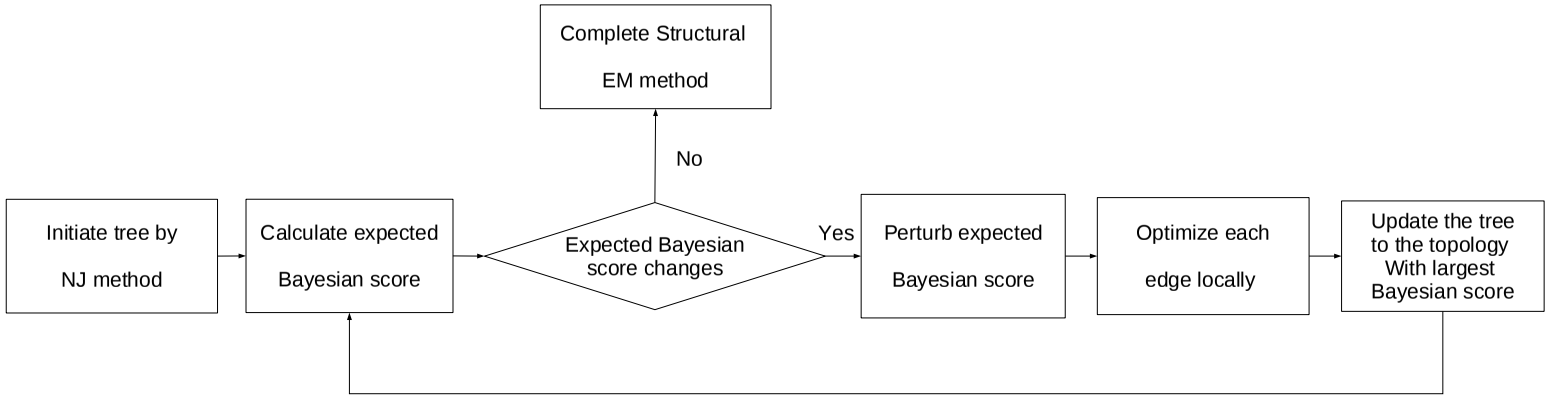



Figure 2: Procedure of structural EM method in phylogenetic tree reconstruction.

In the case of the complete data, the decomposition of the likelihood function to $w_{i,j}(t_{i,j})$ simplifies the structure optimization to two steps. First, we maximize the weights locally. Then we find the structure with the largest sum of weights. However, we usually only know the sequence data for leaves in phylogenetics. Similarly as the standard structural EM, the expected Bayesian score can be calculated as

$$Q(\mathbf{T}, \mathbf{t}; \mathbf{T}_l, \mathbf{t}_l) = E[\log P(\mathbf{O}, \mathbf{H}|\mathbf{T}, \mathbf{t})|\mathbf{T}_l, \mathbf{t}_k] = E[C + \sum_{(i,j)\in T} w_{i,j}(t_{i,j})], \tag{49}$$

which can be reformulated as

$$\begin{aligned} &Q(\mathbf{T}, \mathbf{t}; \mathbf{T}_l, \mathbf{t}_l) \\ &= E[\sum_{(i,j)\in\mathbf{T}} \sum_{(a,b)\in\Sigma^2} N(x_i = a, x_j = b)(\log P_{a\to b}(t_{i,j}) - \log P_b) + C|\mathbf{T}_l, \mathbf{t}_l] \\ &= \sum_{(i,j)\in\mathbf{T}} \sum_{(a,b)\in\Sigma^2} E[N(x_i = a, x_j = b)|T_l, t_l](\log P_{a\to b}(t_{i,j}) - \log P_b) + C \ , \end{aligned} \tag{50}$$

where $t_{i,j}$ is the updated time $t$ between nodes $i$ and $j$ in the current step, and the parameter $\mathbf{t}_l$ is the time learned in the previous step $l$. Recall the statement (10) proved in section 2.3, which indicates that improving the expected Bayesian score leads to the improvement of the real Bayesian score. The expected Bayesian score can also be decomposed to the sum of the weights between linked nodes. Therefore, the best expected Bayesian score can be solved by first optimizing the weights locally with the best edge lengths of all possible pairs of nodes, and then finding the tree structure that combines the highest overall edge weights. In the following sections, we first review the message passing algorithm for efficiently calculating the expected Bayesian score. After that, we discuss the details of how to apply the structural EM method for reconstructing phylogenetic trees.

### 3.3.2 Message passing

Recall that the expected Bayesian score can be represented as

$$Q(\mathbf{T}, \mathbf{t} : \mathbf{T}_l, \mathbf{t}_l) = \sum_{(i,j)\in\mathbf{T}} \sum_{(a,b)\in\Sigma^2} E[N(x_i = a, x_j = b)(\log P_{a\to b}(t_{i,j}) - \log P_b)|\mathbf{T}_l, \mathbf{t}_l] + C. \tag{51}$$

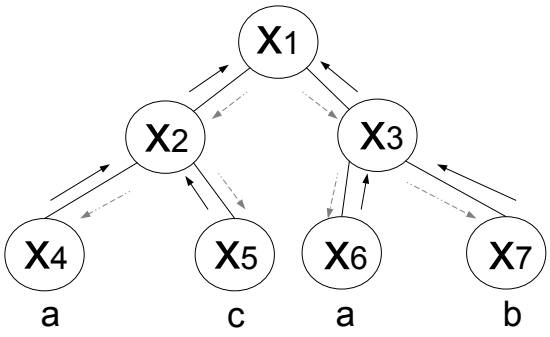



Figure 3: A tree with seven nodes. The character states of the nodes are noted beside them. The black arrows show the order of updating the upward $U$ and $u$ messages. The dashed gray arrows show the order of updating the downward $U$ and $u$ messages.

The first term can be represented as

$$\begin{aligned} &\sum_{(i,j)\in\mathbf{T}} \sum_{(a,b)\in\Sigma^2} E[N(x_i = a, x_j = b)(\log P_{a\to b}(t_{i,j}) - \log P_b)|\mathbf{T}_l, \mathbf{t}_l] \\ = &\sum_{(i,j)\in\mathbf{T}} \sum_{(a,b)\in\Sigma^2} \{[P(x_i = a, x_j = b|\mathbf{T}_l, \mathbf{t}_l)] (\log P_{a\to b}(t_{i,j}) - \log P_b)\}. \end{aligned} \tag{52}$$

Thus, we only need to calculate $P(x_i^k = a, x_j^k = b|\mathbf{T}_l, \mathbf{t}_l)$ locally for each edge, then sum up for all edges and symbols. A naive way for calculating $P(x_i^k = a, x_j^k = b|\mathbf{T}_l, \mathbf{t}_l)$ can be summing up all possible configurations of other nodes. For more efficiently calculating this conditional probability, we review the dynamic programing method based on the message passing algorithm, which was developed by Friedman [18]. For each edge $(i, j) \in T$, denote a subtree including node $i$ as $s(i)$, and

$$\begin{cases} U_{i\to j}(a) = P(s(j)|x_i = a, \mathbf{T}, \mathbf{t}), \\ u_{i\to j}(a) = P(s(i)|x_j = a, \mathbf{T}, \mathbf{t}). \end{cases} \tag{53}$$

Then, starting from leaf nodes, the $U$ and $u$ messages are passed upwards until they reach the root. Initialize $U_{i\to j}(a)$ as:

$$U_{i\to j}(a) = 1\{x_i = a\}. \tag{54}$$

Then we can calculate $U$ and $u$ upwards repeatedly by

$$\begin{cases} u_{i\to j}(a) = \sum_{b\in\Sigma} P_{a\to b}(t_{i,j}) U_{i\to j}(b), \\ U_{i\to j}(a) = \prod_{v\neq j,(v,i)\in\mathbf{T}} u_{v\to i}(a). \end{cases} \tag{55}$$

After the last $u$ message reaches root, the messages are passed down and calculated in same way as they are passed up. In this way, the $U$ and $u$ messages are passed through all nodes both upwards and downwards. Figure 3 is an example of how to update $U$ and $u$ messages in a tree. The character states of the nodes are noted beside them. The arrows show the order of updating the $U$ and $u$ messages. Assuming the space of the characters as $\Sigma = \{a, b, c\}$, we can update the $U$ and $u$ messages in the following steps:

- Calculate the $U$ messages from $x_4$ to $x_2$. Initialize

$$\begin{cases} U_{x_4\to x_2}(a) = 1, \\ U_{x_4\to x_2}(b) = 0, \text{ and} \\ U_{x_4\to x_2}(c) = 0. \end{cases} \tag{56}$$

  Then the $u$ messages from $x_4$ to $x_2$ is

$$\begin{cases} u_{x_4\to x_2}(a) = \sum_{d\in\Sigma} P_{a\to d}(t_{x_4,x_2}) U_{x_4\to x_2}(d) = P_{a\to a}(t_{x_4,x_2}), \\ u_{x_4\to x_2}(b) = \sum_{d\in\Sigma} P_{b\to d}(t_{x_4,x_2}) U_{x_4\to x_2}(d) = P_{b\to a}(t_{x_4,x_2}), \text{ and} \\ u_{x_4\to x_2}(c) = \sum_{d\in\Sigma} P_{c\to d}(t_{x_4,x_2}) U_{x_4\to x_2}(d) = P_{c\to a}(t_{x_4,x_2}). \end{cases} \tag{57}$$

  Similarly, we can calculate $\{U_{x_5\to x_2},\ u_{x_5\to x_2}\}$, $\{U_{x_6\to x_3},\ u_{x_6\to x_3}\}$, and $\{U_{x_7\to x_3},\ u_{x_7\to x_3}\}$ orderly.

- Calculate $U_{x_2\to x_1}$:

$$\begin{cases} U_{x_2\to x_1}(a) = \prod_{v\neq x_1,(v,x_2)\in\mathbf{T}} u_{v\to x_2}(a) = u_{x_4\to x_2}(a) u_{x_5\to x_2}(a), \\ U_{x_2\to x_1}(b) = \prod_{v\neq x_1,(v,x_2)\in\mathbf{T}} u_{v\to x_2}(b) = u_{x_4\to x_2}(b) u_{x_5\to x_2}(b), \text{ and} \\ U_{x_2\to x_1}(c) = \prod_{v\neq x_1,(v,x_2)\in\mathbf{T}} u_{v\to x_2}(c) = u_{x_4\to x_2}(c) u_{x_5\to x_2}(c). \end{cases} \tag{58}$$

  Then we can update $u_{x_2\to x_1}$. After that, we can infer $\{U_{x_3\to x_1},\ u_{x_3\to x_1}\}$ in similar way.

- After $U$ and $u$ messages reach root $x_1$, we pass them down from the root. First, we update $\{U_{x_1\to x_2},\ u_{x_1\to x_2}\}$ by

$$\begin{cases} U_{x_1\to x_2}(a) = \prod\limits_{v\neq x_2,(v,x_1)\in\mathbf{T}} u_{v\to x_2}(a) = u_{x_3\to x_1}(a), \\ U_{x_1\to x_2}(b) = \prod\limits_{v\neq x_2,(v,x_1)\in\mathbf{T}} u_{v\to x_2}(b) = u_{x_3\to x_1}(b), \text{ and} \\ U_{x_1\to x_2}(c) = \prod\limits_{v\neq x_2,(v,x_1)\in\mathbf{T}} u_{v\to x_2}(c) = u_{x_3\to x_1}(c). \end{cases} \tag{59}$$

  Then,

$$\begin{cases} u_{x_1\to x_2}(a) = \sum\limits_{d\in\Sigma} P_{a\to d}(t_{x_1,x_2})U_{x_1\to x_2}(d), \\ u_{x_1\to x_2}(b) = \sum\limits_{d\in\Sigma} P_{b\to d}(t_{x_1,x_2})U_{x_1\to x_2}(d), \text{ and} \\ u_{x_1\to x_2}(c) = \sum\limits_{d\in\Sigma} P_{c\to d}(t_{x_1,x_2})U_{x_1\to x_2}(d). \end{cases} \tag{60}$$

  In the same manner, we can update the following messages orderly until the messages from both directions are obtained: $\{U_{x_1\to x_3},\ u_{x_1\to x_3}\}$, $\{U_{x_2\to x_4},\ u_{x_2\to x_4}\}$, $\{U_{x_2\to x_5},\ u_{x_2\to x_5}\}$, $\{U_{x_3\to x_6},\ u_{x_3\to x_6}\}$, and $\{U_{x_3\to x_7},\ u_{x_3\to x_7}\}$.

After calculating the $U$ and $u$ messages, the probability associated with a node, as well as the probability of an edge can be inferred by

$$\begin{cases} P(x_i = a|\mathbf{T},\mathbf{t}) = \dfrac{P_a U_{i\to j}(a)u_{j\to i}(a)}{\sum_{d\in\Sigma} U_{i\to j}(d)u_{j\to i}(d)P_a}, \\ P(x_i = a, x_j = b|\mathbf{T},\mathbf{t}) = \dfrac{P_a U_{i\to j}(a)P_{a\to b}(t_{i,j})U_{j\to i}(b)}{\sum_{d\in\Sigma} U_{i\to j}(cd)u_{j\to i}(d)P_a}. \end{cases} \tag{61}$$

However, the probability of node pairs that are not directly linked are still left to be computed. Assuming that node $v$ is in the middle of node $i$ and node $j$, by Bayesian theorem, this probability can be calculated by summing up the marginal probability of three nodes as

$$P(x_i, x_j|\mathbf{T},\mathbf{t}) = \sum_{x_v} \frac{P(x_i, x_v|\mathbf{T},\mathbf{t})P(x_v, x_j|\mathbf{T},\mathbf{t})}{P(x_v|\mathbf{T},\mathbf{t})}. \tag{62}$$

To update the probabilities of all node pairs, we can use the following strategy. First, a pool of nodes between which the probabilities of all possible pairs are already calculated is created. Pick one leaf node as the initial element in the pool. Then nodes which are not in the pool are gradually added to the pool in the way that the new node added is the neighbor of one of the elements in the pool. After introducing

a new node to the pool, the probability between this new node and other nodes in the pool, if not known, can be calculated by using its neighbor as a bridge as shown in Equation (62). Finally, we have all of nodes added to the pool, thus finish computing the probabilities of all node pairs. Using the example in Figure 3, we can obtain the probabilities of all node pairs in the following steps

- Initialize the pool with $\{x_1, x_2\}$.
- Pick $x_1$ from the pool. Add one of the neighbors of $x_1$, if not yet in the pool, to the pool. Here $x_3$ is added. Update $P(x_2, x_3|\mathbf{T}, \mathbf{t})$ by using $x_1$ as the bridge.
- When all the neighbors of $x_1$ are added in the pool, randomly pick another node which was not picked before. Here we pick $x_2$. First, we add $x_4$, one of the neighbors of $x_2$ into the pool. We can update $P(x_4, x_1|\mathbf{T}, \mathbf{t})$ and $P(x_4, x_3|\mathbf{T}, \mathbf{t})$ by using $x_1$ as the bridge. Then we add $x_5$ to the pool, and update $P(x_5, x_4|\mathbf{T}, \mathbf{t})$, $P(x_5, x_3|\mathbf{T}, \mathbf{t})$ and $P(x_5, x_1|\mathbf{T}, \mathbf{t})$.
- In the same manner, we pick $x_3$ and add its neighbors ($x_6$ and $x_7$) and update the respective probabilities. When all the nodes in the tree are added in the pool, we complete the process of obtaining the probabilities of all node pairs.

The dynamic programming method computes the expectation counts using $O(M \cdot N \cdot |\Sigma|^3)$ time. When the number of sequences and lengths of sequences are large enough, the time requirement becomes extremely prohibitive. One possible way to relieve the problem is to approximate the probabilities [18] by

$$P(x_i, x_j|\mathbf{T}, \mathbf{t}) = P(x_i|\mathbf{T}, \mathbf{t})P(x_j|\mathbf{T}, \mathbf{t}). \tag{63}$$

As a result, the running time can be decreased to $O(M \cdot N \cdot |\Sigma|)$. Note that the node pairs linked directly by the edges are still calculated using $U$ and $u$ messages. This approximation assumes that the nodes which are not directly linked are independent. To make the approximation more reasonable, we can confine that only the nodes that have more than certain number of nodes, for example three, in their shortest path can be calculated by the approximation. On the other hand, the nodes that are closer to each other, which are supposed to have stronger dependence between each other, are still calculated by (62).

### 3.3.3 Structural EM iterations

Now all the components for the structural EM application for reconstructing phylogenetic trees are ready. In the E-step, the expected Bayesian score is calculated locally for each pair of nodes based on previous learned topology $\mathbf{T}_l$ and edge length $\mathbf{t}_l$. Unlike in the standard structural EM method, the local expected Bayesian scores are calculated on unlinked node pairs besides the linked pairs. By extending the calculation of the Bayesian scores to all possible node pairs, we can utilize a more efficient algorithm, the maximum spanning tree algorithm [18], to update the tree structure.

The M-step is performed in two steps. First, the link lengths between all node pairs are optimized to maximize the local Bayesian scores. Note that the constant in the Bayesian score function can be omitted in the calculation process. As a result a weight matrix of $(2N-2) \times (2N-2)$ is built. In the second step, the maximum spanning tree algorithm is performed to search through the matrix to find a combination of linked nodes which have the maximum overall weights. The maximum spanning tree algorithm finds a topology with weights equal or more than all other possible spanning trees in time $O(N(E) \log N(V)$, where $N(E) = 2N-2$ is the number of edges and $N(V) = 2N-4$ is the number of vertices. Compared to the one step updating strategy, the maximum spanning tree algorithm speeds up the process by finding a more reasonable structure in each iteration. In the structural EM method, most of the computing time is used for calculating the expected Bayesian scores and optimizing them locally. To find a reasonable structure, if we only change a small part of edges randomly in M-step, it will usually need more iterations than the maximum spanning tree algorithm, thus more time for completing the whole structural EM search. Moreover, the maximum spanning tree algorithm can find the topology that gives the best lower bound for the improvement of real Bayesian score each time.

Another modification to the standard structural EM method is that in the initial step, a better guess of the tree structure is made. This is done by initiating the phylogenetic tree by neighbor-joining (NJ) method [54]. Neighbor-joining method reconstructs phylogenetic trees on the basis of the distances between sequences. The distances between sequences can be calculated by the minimum changes which are needed to transform one sequence to another. In the neighbor-joining method, each time the closest nodes are merged until all nodes are merged to form the root. Based on a reasonable starting structure, the structural EM method can converge

faster and avoid being stuck in spurious structures. Another important issue in the phylogenetic tree reconstruction is that the tree structure is assumed to be a binary tree. To accomplish this requirement, we can transform a tree to its equivalent binary tree which has the same probability. We discuss this process in detail in section 3.3.4. Thus, the procedure of the structural EM method for reconstructing phylogenetic trees can be summarized as:

1: Initialize a phylogenetic tree by the neighbor-joining method.
2: **repeat**
3: For each node pair, compute the local expected Bayesian score based on the topology $\mathbf{T}_l$ and the length $\mathbf{t}_l$ inferred in the previous step in the way described in section 3.3.2. {E-step}
4: Find the link length which maximizes the expected Bayesian score locally by

$$\begin{aligned} t_{i,j} &= \arg\max_{t_{i,j}} E[w_{i,j}(t_{i,j})|\mathbf{T}_l, \mathbf{t}_l] \\ &= \arg\max_{t_{i,j}} \left\{ \left[ \sum_{k=1}^{m} P(x_j^k = a, x_j^k = b)|\mathbf{T}_l, \mathbf{t}_l \right] (\log P_{a\to b}(t_{i,j}) - \log P_b) \right\}. \end{aligned} \tag{64}$$

Build a matrix with the maximized local expected Bayesian score between all node pairs. Update the topology $\mathbf{T}_{i+1}$ using the maximum spanning tree algorithm. Transform the resulting tree to a binary tree with same probability. {M-step}
5: **until** The expected Bayesian score converged, or run out of time..

When we have a set of sequences:

A: ACCCTGC
B: TCCCTGC
C: ACCTTGC
D: ACCTTGA
E: ATCCATC
F: ATTCATC
G: ACCCCTC
H: ATTCATT

we have the results:

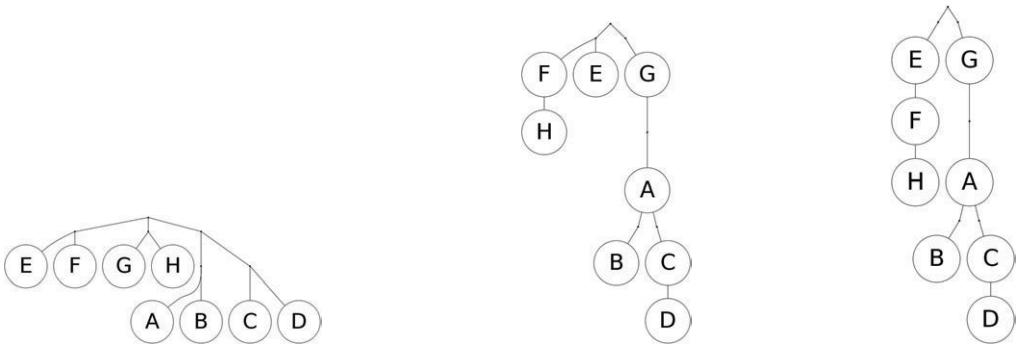



Figure 4: The trees learned by the structural EM method for the sequence data described above. In this example, no noise was added in the learning process. The structural EM method converged in the second run.
(a) The initial tree. (b) The tree learned in the first run. (c) The tree learned in the second run. The structural EM method converged in this run.

The steps of how the structural EM method reconstructed the phylogenetic tree for the data are shown in Figure 4. In this example, no noise was added in the learning process. The structural EM method converged in the second run.

### 3.3.4 Transforming a tree to an equivalent binary tree

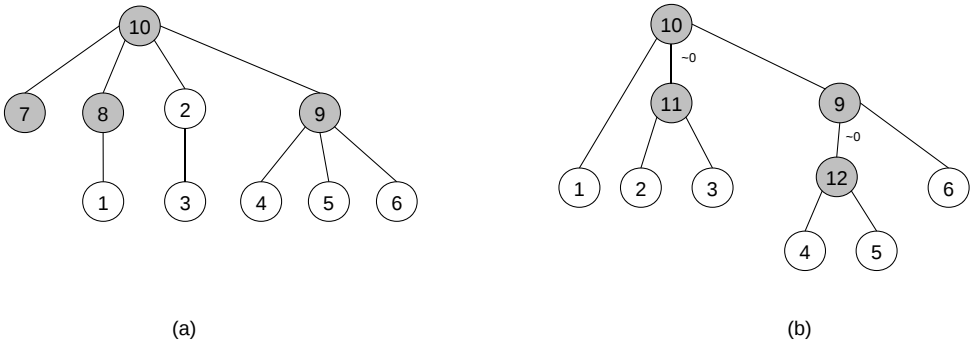



Figure 5: An example of how to transform an arbitrary tree to a binary tree. The figure is adapted from [17].

In phylogenetic analysis, it assumes that all divergence events of two species bifurcate from the ancestors.In phylogenetics, the binary tree requirement implies that the observed nodes have only one neighbor and the hidden nodes have three neighbors. The structural EM method for phylogenetic analysis first finds the structure that are not confined to be a binary tree by the maximum spanning tree algorithm. Then it transforms the structure to its equivalent binary tree which has the same probability. To do this, some nodes are removed and new ones are added elsewhere to gradually approach the binary structure. On three conditions, adding or removing a node does not change the probability of the tree:

- Condition 1: When a hidden node has only one neighbor, it can be removed without changing the tree probability.
- Condition 2: When a hidden node has two neighbors, it can be removed without changing the tree probability. Its two neighbors are then linked together, with the new edge length which is the sum of the edge lengths between the removed node and the two neighbors.
- Condition 3: When a hidden node has more than three neighbors or an observed node has more than one neighbor, a new hidden node can be added as its child. The distance between the new node and the node with more neighbors than constrained is set to be zero.

For condition 3, in practice, the zero edge length is set to a small number which is larger than zero for the separation of the two nodes. The new node is put in the position as the parent of two of the neighbors, while the edge lengths are the same as the node to these two neighbors. Heuristically, the neighbors which are closest to each other are grouped together first.

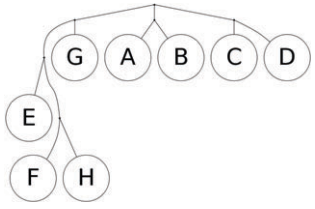



Figure 6: The equivalent binary tree of the tree in Figure 4c.

Figure 5 is an example of how to transform an arbitrary tree to a binary tree. The Observed nodes are 1 to 6 which are colored in white. Others are hidden nodes and are colored in gray. Node 7 is a hidden node and has only one neighbor: node 10. It is removed under the condition 1. Node 8 has only two neighbors and is a hidden node. Thus it is deleted under the condition 2. Based on the condition 3, node 11 and node 12 are inserted to reduce the degree of nodes that have more neighbors than constrained. Node 11 is added to be the parent of node 2 and node 3, with an edge of length zero linked to node 3. Then node 12 is inserted as the parent of node 4 and node 5, with an edge of length zero to node 9. Node 4 and node 5 are grouped together because they are closer to each other than other possible pairs between node 4, node 5 and node 6. In the example of the structural EM method for learning a set of DNA sequences in section3.3, we can transform the tree in Figure 4c as the binary tree in Figure 6.

Here we prove the correctness of the three cases. For the first case, when a hidden node $j$ has only one neighbor, it is in the leaf position of a tree. Pick $j$ as the root. Based on Equation (42), reform the formula for calculating the marginal probability of the tree conditioned on the observed nodes:

$$\begin{aligned} P(x_1,...,x_N|\mathbf{T},\mathbf{t}) &= \sum_{\{x_k|k\neq j,N<k<2N-2\}} P(x_k|\mathbf{T}',\mathbf{t}') \sum_{x_j=b,b\in\Sigma} P_{a\to b}(t_{i,j}) \\ &= \sum_{\{x_k|k\neq j,N<k<2N-2\}} P(x_k|\mathbf{T}',\mathbf{t}'), \end{aligned} \quad , \tag{65}$$

where $\mathbf{T}'$, and $\mathbf{t}'$ represent the new tree structure without hidden node $j$. From the above equation, it is obvious that the new tree has the same marginal probability as the old one.

In the second condition, when a hidden node $j$ is in the middle of two nodes, it can be deleted and the nodes in its two ends, denoted as $i$ and $k$, are linked together. The length of the new edge is the sum of the removed edges. The second case can be proven by the Chapman-Kolmogorov equation locally:

$$P(x_i,x_k|t_{i,j}+t_{j,k}) = \sum_{x_j} P(x_i,x_j|t_{i,j})P(x_j,x_k|t_{j,k}). \tag{66}$$

The last case can be proven in the similar way as the second case. Denote the new node as $k$, the node with more neighbors than constrained is $i$, and one of its neighbors as $j$. After the reformation, $j$ is changed as the child of $k$. Then we have:

$$P(x_i,x_j|t_{i,j}) = \sum_{x_k} P(x_i,x_k|t_{i,k})P(x_k,x_j|t_{k,j}). \tag{67}$$

This is because when $t_{i,k} \to 0$, and $t_{k,j} = t_{i,j}$, we have $t_{i,j} \approx t_{i,k} + t_{k,j}$.

The deletion and insertion steps can be carried out in an arbitrary order. One possible way is to go through all nodes and performing the deletion step and the insertion step iteratively, while the inserted nodes can be only selected from the deleted nodes [18]. However this strategy needs more iterations because in insertion steps there may be not enough deleted nodes to reuse. Therefore, another run of the deletion step is needed to free new hidden nodes. To decrease the number of iterations, when there is no hidden node available for insertions, new hidden nodes can be created and added. It allows as many as possible insertions in one iteration and thus does not need to wait for an extra run of the deletion step.

### 3.3.5 Avoiding local optima

As discussed in section 2.3, the structural EM method is a greedy method and does not guarantee to find the global optimum. A straightforward solution is to try the structural EM method from different random starting points and select the best scored result. But this process can be extremely time consuming. Furthermore, it is unknown how many runs are enough to find the nearly global optimum. Instead of the brute-force strategy, extra information can be incorporated to guide the structural EM searching process. This is why the neighbor-joining method, which generates a reasonable tree which is likely to be close to the global optimum in the model space, is used for constructing the initial tree. However, in the learning process, a previous selected model can make the learned parameters favor itself. Thus, the learned tree structure can be biased to the previous one, which is another cause for resulting in a sub-optimal tree.

To address this problem, the simulated annealing (SA) method [24] can be utilized to help the structural EM searching process to escape from local optima. The simulated annealing method is inspired by the annealing in metallurgy. If the temperature decreases in the process of the cooling of metal is very slow, the atoms will have more chance of wandering randomly in the state space and thus avoid being stuck in their local states. Similar to this process, the SA method introduces larger randomnesses in the begining stage of the structural EM searching process, which allows more freedom to escape from local optima. Gradually, the temperature is decreased to make the randomness smaller. In the final stage, when the introduced randomness is very small, the structural EM method then searches purely in a hill-climbing manner and converges.

One way of incorporating the SA method to the structural EM learning is to make a random disturbance of the learned tree structure [2, 8]. The topology can be changed randomly by swapping node pairs in the quartets of the topology [62], or by one step of the subtree pruning and regrafting [62]. Instead of randomizing the topology directly, another way of incorporating the SA method is by adding randomnesses to the expected Bayesian score, which is the guide for learning the topology. In the following part of this section, we discuss two ways under the simulated annealing method framework: perturbing edge weights and perturbing position weights.

In the first way, the edge weight matrix, which is used for constructing the maximum spanning tree, is perturbed by

$$\widetilde{w}_{i,j}(t_{i,j}) = w_{i,j}(t_{i,j}) + \epsilon_{i,j}, \tag{68}$$

where $\epsilon_{i,j}$ is sampled from the normal distribution with mean zero and variance $\delta^2$. To preserve the symmetry of the matrix, the noises follow

$$\epsilon_{i,j} = \epsilon_{j,i}. \tag{69}$$

The variance is decreased by multiplying a factor $\rho < 1$ each time. Denote the maximum spanning tree learned from the original matrix is $T$. Tree $T'$ is obtained by making a modification to one of the edges of $T$. Assume the original edge between node $i$ and node $j$ are modified to the edge between $i'$ and $j'$. In the perturbed matrix, their lengths are perturbed by $\epsilon_{i,j}$ and $\epsilon_{i',j'}$ respectively. Then,

$$\widetilde{W}[T] - \widetilde{W}[T'] = W[T] - W[T'] + \epsilon_{i,j} - \epsilon_{i',j'}. \tag{70}$$

Because each $\epsilon$ is drawn independently from a Gaussian distribution, the probability of moving from $T$ to $T'$ is:

$$P(\widetilde{W}[T'] > \widetilde{W}[T]) = \Phi(\frac{W[T'] - W[T]}{\sqrt{2}\sigma}), \tag{71}$$

where $\Phi(x)$ is the Gaussian cumulative function. By decreasing $\sigma$, $P(\widetilde{W}[T'] > \widetilde{W}[T])$ is decreased. In other words, when the variance of noises is decreased, the probability of moving from the maximum spanning tree structure to some other structure is decreased. By decreasing $\sigma$ gradually, the structural EM method starts from moving through the searching space with more freedom; then, as the temperature decreases towards zero, the it slowly converges to an optimum which is hopefully the global one. Therefore, the structural EM method with perturbed weights only modifies the E-step as:

- **E-step:** For each node pair, compute locally the expected Bayesian score based on topology $\mathbf{T}_l$ and length $\mathbf{t}_l$ inferred in the previous step in the way described in section 3.3.2. Perturb the weight matrix by adding the Gaussian noise to each element in the matrix. The noises are sampled independently from the Gaussian distribution with variance $\sigma$. Decrease $\sigma$ by $\sigma_{l+1} = \sigma_l * \rho$ each time, where $\rho < 1$.

The second way of incorporating the SA method is to assign random weights to different positions of the sequences. This method is inspired by the bootstrapping method which is widely used in phylogenetic analysis [28] to avoid the over-fitting of training data. Formally, the resampling in the bootstrapping method is performed by selecting random samples from the data and repeating the process for enough times. The resampling strategy can also be regarded as assigning weight to each position randomly from the set of $\{1, 0\}$. Therefore, the process of assigning random weight to each position mimics the general bootstrapping process. This process can be formulated as

$$\widehat{w}_{i,j} = \sum_{k=1}^{m} \sum_{(a,b)\in\Sigma} \omega_k 1\{x_i^k = a, x_j^k = b\}[\log P_{a\to b}(t_{i,j}) - \log P_b]. \tag{72}$$

In each iteration, the position weight $\omega_k$ is sampled independently from the Gamma distribution with variance $\sigma$ that decreases each time by multiplying $\rho$ ($\rho < 1$). The E-step of the structural EM method can then be modified as

- **E-step:** For each node pair, compute the local expected Bayesian score based on topology $\mathbf{T}_l$ and length $\mathbf{t}_l$ inferred in the previous step $l$ in the way described in section 3.3.2. Sample the position weight randomly from Gamma distribution, whose variance is decreased by $\rho$ ($\rho < 1$) each time. Assign the position weight as described in Equation (72).

## 3.4 Experiments

In this section, we first review other commonly used computational methods in phylogenetics. Then, we illustrate two experiments of the structural EM method in phylogenetic analysis. The first experiment was performed on nine sets of artificially generated data. In the second experiment, we applied the structural EM method in analyzing the regulatory genes in *streptococcus pneumoniae*.

### 3.4.1 Other computational methods in phylogenetics

We compare the structural EM method with five different computational methods from three commonly used programs (PAUP, PHYLIP and MrBayes) in the experiment of evaluating their performance in phylogenetic analysis. The five computational methods include the neighbor-joining method [54], the least-squares method [13], the maximum parsimony method [12], the maximum likelihood method using heuristic algorithms [62], and the maximum a posteriori method using simulation algorithms [50]. The first four methods are applied both PHYLIP and PAUP. The simulation method for the Bayesian analysis of phylogenetics, is utilized by the MrBayes program [50]. The following paragraph gives a short introduction to these methods.

The first two methods, the neighbor-joining method and the least-squares method, are based on the distances between sequences. The distance between two sequences measures the differences between them. The neighbor-joining method reconstructs the tree structure by iteratively joining the nearest nodes until reaching the root. The least-squares method uses the Fitch-Margoliash criterion [13] to fit branch lengths to each topology, and selects the topology with the smallest sum of squared errors. The minimun-evolution [31, 53] method is similar to the least-square method. It also uses the Fitch-Margoliash criterion to fit branch lengths to topologies, but it chooses the topology with the smallest sum of branch lengths rather than by the goodness of the fitting of the sum of square errors. The third method, the maximum parsimony method aims to find the tree structure that generates the sequences through the path of least substitutions. In the fourth method, which usespl the heuristic search for reconstructing the maximum likelihood tree, the species are added to the topology successively until all of them are added [9]. Each time the tree topology with the highest likelihood is selected. Additionally, before the next species is added, local rearrangements of the existing topology are performed and accepted only if the rearrangement increases the probability of the tree. The process completes when all the species are added and no improvement can be made by local rearrangements. The last method uses simulation techniques, the Markov chain Monte Carlo (MCMC) [47] method, to approximate the posterior likelihood of the phylogenetic trees for the sequence data. It samples from a Markov chain that has the posterior distribution of the phylogenetic trees as its equilibrium distribution. Finally, it generates the set of trees with highest probabilities, and can provide the consensus tree or trees representing them.

### 3.4.2 Analyzing artificial data in phylogenetics

In this experiment, nine sets of data were generated by INDELible [14]. The INDELible program is able to generate aligned DNA sequences including insertions and deletions under various models. First, we need to set the parameters for the *generaters* of CTMC, $\varrho$, to define the model in INDELible. Then, the substitution rate matrix is $e^{\varrho t}$. Then, the insertion rate and the deletion rate need to be set as per substitution rate. See [14] for details about how to set the parameters for INDELible.

In the experiment, the JC model, the F81 model and the K2P model were used for generating nine data sets, with each model used for three data sets. All the data sets used the same parameters for the insertion rate and the deletion rate. For instance, the insertion rate was 0.04 and the deletion rate was 0.08. Each data set had 100 sequences with the length of sequence in the root as 500. Data sets 1 to 3 used the JC model with $\alpha = 1$. The F81 model was used for data sets 4 to 6. In the F81 model, the probabilities of different nucleotides were set as:

$$\begin{cases} P(T) = 0.4, \\ P(C) = 0.3, \\ P(A) = 0.2, \text{ and} \\ P(G) = 0.1 \end{cases} \tag{73}$$

Data sets 7 to 9 were simulated by the K2P model. In the K2P model, the transition/transversion ratio was set to be 1.25.

In the experiment, the programs used the same parameters, if required, for learning the phylogenetic trees as those used for the data simulation. The qualities of the results were measured by the average sign similarities. The higher the average sign similarity, the better the result. See Section 4.2.2 for the details about the average sign similarity.

In all the results, the structural EM method obtained average sign similarities of about 80%, which were consistent for all the data sets. However, the other methods except MrBayes were generally better and could give as high as more than 90% of the average sign similarities. The results showed the strength of the traditional phylogenetic methods, especially the parsimony method and the maximum likelihood method using heuristic algorithms. These two methods usually obtained the best results for the nine data sets. On the other hand, the structural EM method, which is also based on the Bayesian theory, could not find a structure as good as the

maximum likelihood method using heuristic algorithms. The reason might be the structural EM method's tendency to be stuck in local maxima. MrBayes showed inconsistency in its results. For example, for data set 1 using the JC model, the resulting phylogenetic tree had the average sign similarity as high as 91%. But for data set 4 using the F81 model, the average sign similarity was only 49%. The reason might be that, for a large data set, MrBayes required a remarkably long running time. Therefore, it was very hard to find a good stopping point for convergence. The results also indicated that the same method in different programs could have different results. For example, for data set 1, the result of the parsimony method in PHYLIP generated the result with the average sign similarity which is 2% larger than the result of the parsimony method in PAUP. This could be due to the different search algorithms utilized in different programs. The results are shown in Table 1 and Figure 7.

### 3.4.3 Clustering capsular regulatory genes in *streptococcus pneumoniae*

In this experiment, the structural EM method was used for analyzing the regulatory genes in *streptococcus pneumoniae*. *Streptococcus pneumoniae*, a significant human pathogenic bacterium, is recognized as the major cause of pneumonia. The virulence of *streptococcus pneumoniae* is mainly determined by its capsule type. In the paper by Varvio *et al.* [67], the capsular regulatory genes were analyzed to explain the epidemiological characteristics of the serotypes in *streptococcus pneumoniae*. Varvio *et al.* first aligned the capsular regulatory genes using Clustal program. Then, minimum-evolution phylograms were used for reconstructing the phylogenetic trees from the aligned sequences.

To inspect the performance of the structural EM method on the regulatory genes in *streptococcus pneumoniae*, sequences from four core capsular genes (*wzg*, *wzh*, *wzd*, and *wze*), and two serotype-specific glycosyltransferase genes (*wchA* and *wchF*) were used in our experiment to compare with the results by Varvio *et al.*. The serotype-specific glycosyltransferase genes code the initial glycosyltransferases in different serotypes. They are also important regulatory genes besides the genes which code the capsule. In the experiment by Varvio *et al.*, the capsular genes *wzh*, *wzd* and *wze*, and the glycosyltransferase gene *wchA*, were separated into two clans. Inside each clan, the genes were located closely to each other with approximately equal distances. The two clans are named as the blue clan and the red clan. The capsular genes in the blue clan have GC contents of about 5% higher than the genes in the

Table 1: Average sign similarities of the phylogenetic trees learned by different methods on nine artificial generated phylogenetic data sets.

| Method | Data 1 | Data 2 | Data 3 | Data 4 | Data 5 | Data 6 | Data 7 | Data 8 | Data 9 |
|---|---|---|---|---|---|---|---|---|---|
| Structural EM | 79% | 84% | 83% | 79% | 84% | 82% | 83% | 81% | 85% |
| Neighbor-joining (PHYLIP) | 91% | 91% | 91% | 87% | 91% | 90% | 92% | 87% | 84% |
| Neighbor-joining (PAUP) | 91% | 91% | 91% | 90% | 90% | 90% | 88% | 88% | 88% |
| Least-squares (PHYLIP) | 93% | 93% | 93% | 91% | 90% | 90% | 90% | 89% | 77% |
| Least-squares (PAUP) | 93% | 93% | 93% | 91% | 91% | 91% | 90% | 90% | 90% |
| Maximum-likelihood (PHYLIP) | 93% | 93% | 93% | 93% | 93% | 93% | **95%** | **94%** | 92% |
| Maximum-likelihood (PAUP) | 94% | **95%** | 93% | 93% | 94% | **95%** | 94% | **94%** | **94%** |
| Parsimony (PHYLIP) | **95%** | **95%** | **95%** | **95%** | 92% | 91% | 93% | 92% | 92% |
| Parsimony (PAUP) | 93% | 93% | **95%** | 93% | **95%** | **95%** | **95%** | 93% | 93% |
| MrBayes | 91% | 68% | 74% | 49% | 66% | 82% | 69% | 70% | 87% |

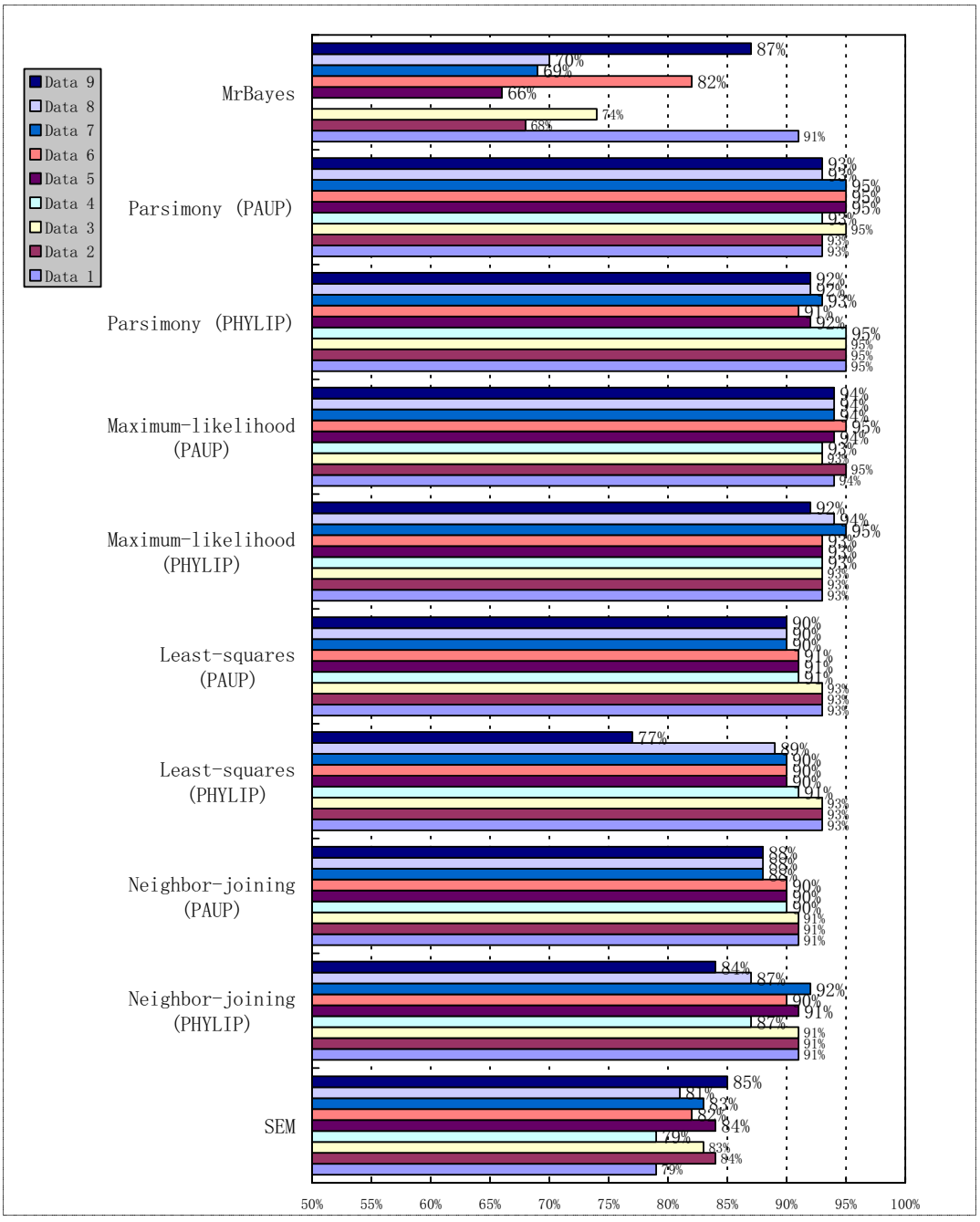
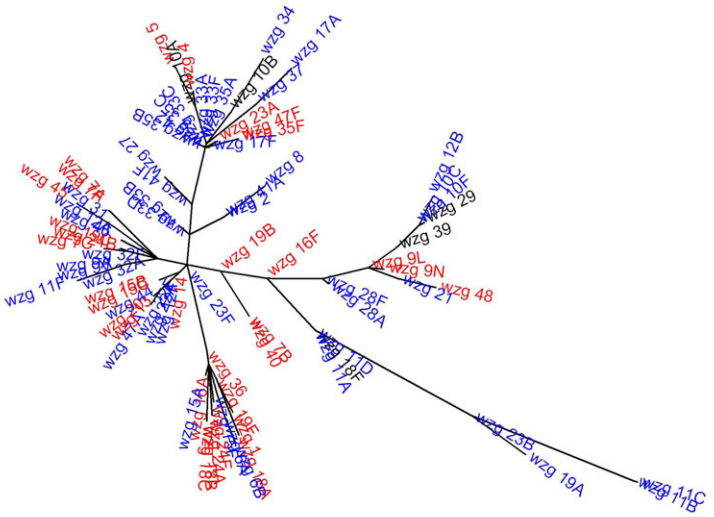
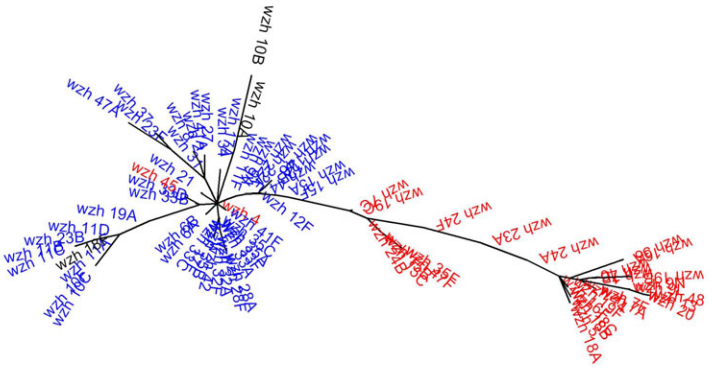


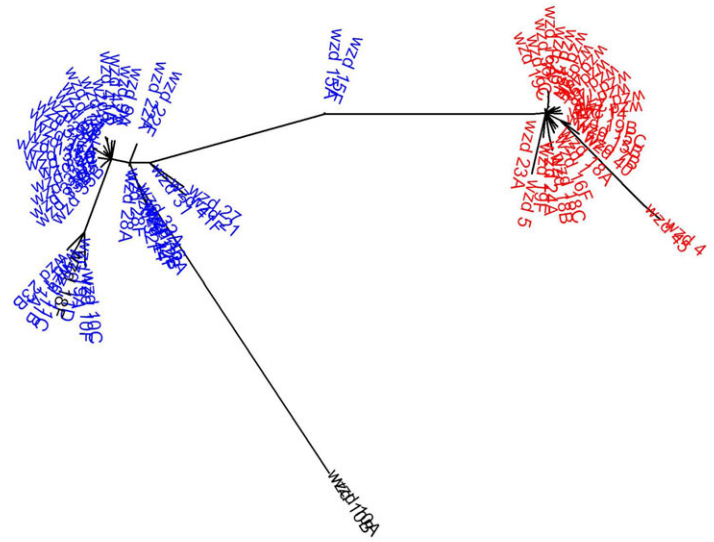

Figure 7: Average sign similarities of the phylogenetic trees learned by different methods on nine artificial generated phylogenetic data sets. The average sign similarities are shown from 50% in the bar chart.



(a) *wzg*.

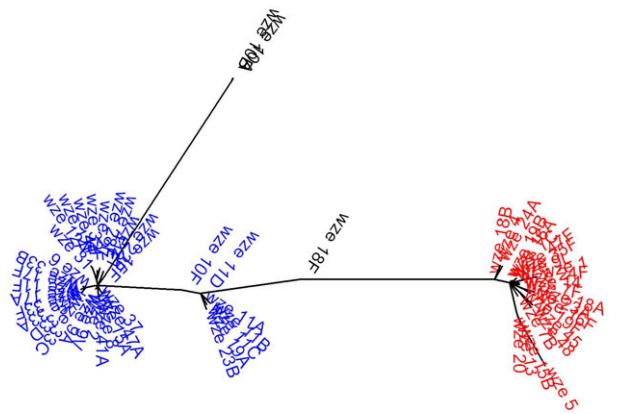


(b) *wzh*.

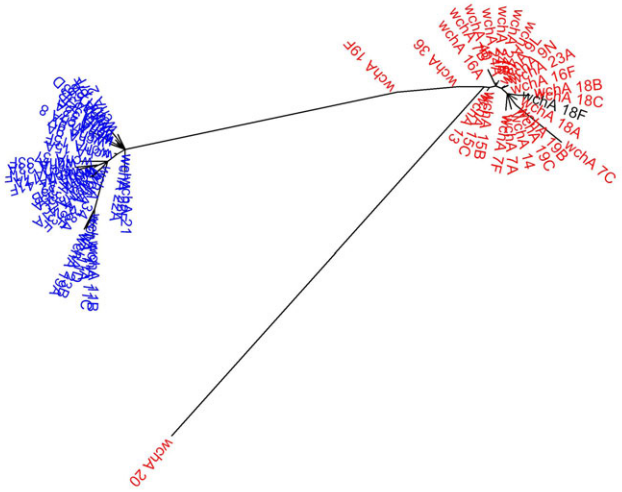


(c) *wzd*.

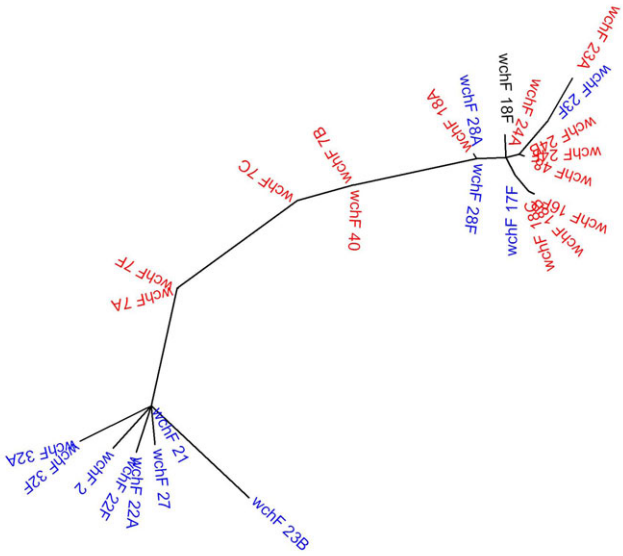
Figure 8: The phylogenetic tree reconstructed by the structural EM method for *wzg*, *wzh*, *wzd* genes of *streptococcus pneumoniae*. The red and blue clan genes are colored respectively in red and blue. The black color represents the genes from ambiguous serotypes.




(a) *wze*.




(b) *wchA*.




(c) *wchF*

Figure 9: The phylogenetic tree reconstructed by the structural EM method for *wze*, *wchA*, and *wchf* genes of *streptococcus pneumoniae*. The coloring scheme is the same as in Figure 8.

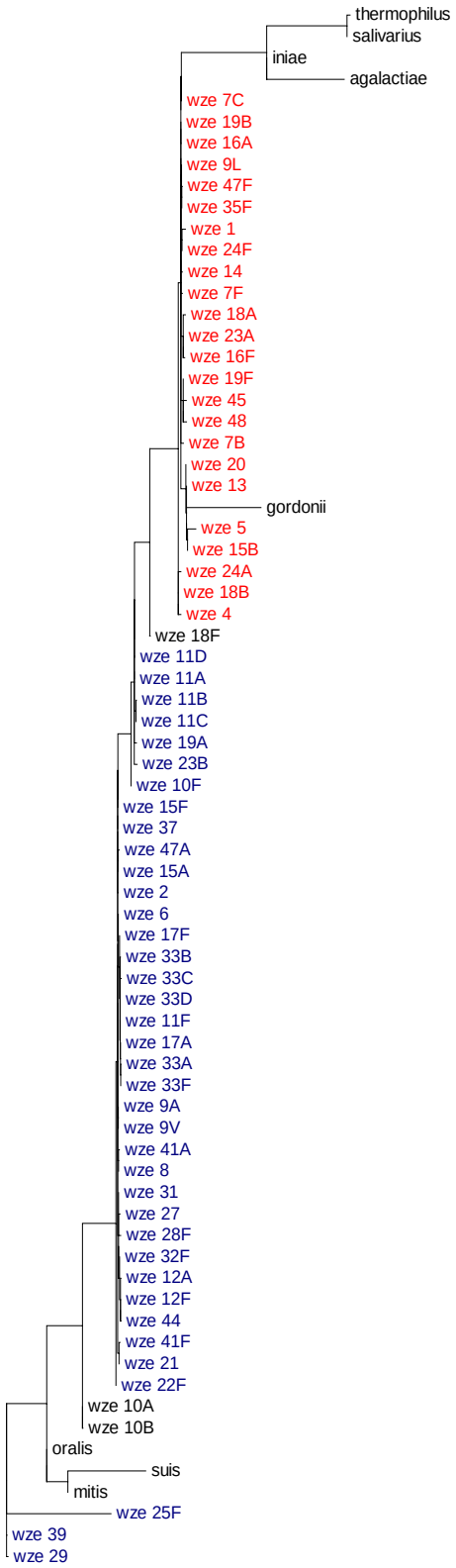



Figure 10: The phylogenetic tree reconstructed by the structural EM method for *wze* genes of *streptococcus pneumoniae* and eight other streptococcus species. The coloring scheme is the same as in Figure 8.

The phylogenetic trees reconstructed by the structural EM method are shown in Figs 8 - 10. Figure 1 in [67] shows the results of the experiment by Varvio *et al.*. Figure 2 in [67] shows the recombination events of the pneumococcal blue and red sequence types in the capsular regulatory genes and glycosyltransferase genes. The point where one sequence type changes to another is the break point of the recombination event.

A further analysis of *wze* genes with more distantly related sequences was performed (Figure 10). To represent the distances between different species more clearly, the phylogenetic tree was depicted in the traditional layout (Figure 10), which was different from the radial layout in Fiigure 9. The *wze* genes from eight other streptococcus species and three distantly related pneumoniae sequences (25F, 29, 39) were added into the data set. The sequences from *S.oralis* and *S.mitis* are closely related to the pneumoniae blue sequence. The *S.gordonii*, *S.thermophilus*, *S.alivarius*, *S.inia*, *S.agalacitiae*, and *S.suis* are more distantly related. The sequences from 39, 29, and 25F do not belong to either red or blue clans. In fact, 25F is most distantly related to other pneumoniae sequences [67]. Among these species, *S.oralis*, *S.mitis*, *S.gordonii* and *S.pneumoniae* belong to the mitis group. *S.thermophilus* and *S.salivarius* are from the salivarius group. *S.inia* and *S.agalacitiae* are members of the pygenic group.

The resulting phylogenetic trees by the structural EM method were similar to the result by Varvio *et al.* except in two cases. First, in the phylogenetic tree for *wzh* genes from the experiment by Varvio *et al.*, genes 23F, 31, and 47A were located in the red clan, but in positions which were close to the blue clan serotypes. However, according to the study of Varvio *et al.*, these three genes are blue clan genes based on their sequence contents (Figure 2 in [67]) and GC contents [67]. Varvio *et al.* also found that not all the genes were grouped to the right clans in the resulting phylogenetic trees. In the phylogenetic tree learned by the structural EM method, genes 23F, 31, and 47A were put deeply inside the blue clan. This result could be more reasonable than the results by Varvio *et al.*. In the second case, it clustered *S.suis* closer to the blue clan, which is contrary to the biological background. However, except these two cases, the resulting phylogenetic trees by the structural EM method are biological consistent.

# 4 Structural EM method in stemmatology

In this chapter, we discuss how to apply the structural EM method in stemmatology. We first introduce the background in stemmatology. Then we discuss how to modify the structural EM method to apply in stemmatology. Finally, we show the results of the structural EM method in three sets of stemmatic data and compare them with other methods.

## 4.1 Introduction to stemmatology

Before the printing was widely used, the manuscripts were usually copied by hand. In the coping process, scribes could introduce modifications to the manuscripts intentionally or by accident. For example, the scribes could alter a word to enhance the meaning or insert some comments in the original manuscripts. On the other hand, some modifications might just be created unintentionally by ignoring or altering some words while copying. But no matter by what means the modifications were introduced, the modified manuscript might be served as the source manuscript and thereby the modifications were propagated and accumulated in the descendant copies. This process resembles the biological process in which genetic mutations are introduced and propagated during the evolution of the species. The evolutionary relationships of manuscripts can also be represented by a family tree, where a node represents a manuscript and an edge represents the relationship of a source manuscript and its direct copy. This family tree, or the *stemma*, is parallel to the phylogenetic tree in biology. As early as in 19th century, Karl Lacnhmann had started to do stemmatic analysis by reconstructing a diagram to show the sharing of the same differences between manuscripts [29], which was used to group manuscripts which were copied from the same source.

Stemmatic analysis aims to recover the relationships of a set of manuscripts, which is analogous to phylogenetic analysis for reconstructing the relationships between species. In this section, we compare the models and processes in these two fields.

In phylogenetic analysis, the family tree is usually assumed to be a binary tree. This is because the separation of more than two species is unlikely to happen at the same time point in the phylogenetic process [58]. As a result, most programs for reconstructing phylogenetic trees are based on the model for reconstructing a binary tree. However, phylogenetic networks can also be used for representing the reticulate events, such as hybridization, horizontal gene transfer, recombination, or

gene duplication, in the evolutionary history in phylogenetics when the binary structure is not suitable. For example, SplitsTree [30] is one of the programs that infer network structures instead of binary trees. On the other hand, in stemmatology more than two copies can be derived from the same source manuscript, thus resulting in polytomies in stemmatic trees. Another difference is that in phylogenetics, the observed species are only located in the leaf nodes; while in stemmatology, the observed manuscripts can be put in the internal nodes. In phylogenetics, the species keep changing all the time. Therefore it is assumed that no ancestral form, which is located in the internal nodes, can be retrieved for analysis. However, in stemmatology, the manuscripts usually stay unchanged once they have been produced. Therefore, the ancestral manuscripts can be retrieved and should be represented as the internal nodes in the stemmatic tree.

On the other hand, there are several processes in stemmatology that have analogies in phylogenetics. For instance, the processes of deletion, insertion, and changing of the order of words are analogous to the processes which happen on nucleotides or amino acids in phylogenetics. Furthermore, in stemmatology, there are also processes which are parallel to the processes which lead to the incongruities between the gene tree and the species tree in phylogenetic analysis. We discussed these processes in phylogenetics in Section 3.1. In stemmatology, the similar processes include three types. First, the exemplar shift is analogous to the gene recombination in phylogenetics. One possible cause for changing of exemplars in the copying process is that a scribe may consider that for certain part of the manuscript, another version of the source is more reliable. Thus, when different parts of manuscripts are used as training data for stemmatic analysis, the reconstructed stemmatic trees can be different. One example is the different trees built based on the first and second half of manuscripts the of *Wife of Bath's Tale* [29].

Second, the lateral transfer is also observed in stemmatology, when some parts of documents are copied from totally different genre of documents rather than merely from another version of the same document. For example, a poem for *Kings of England II* shows the strong evidence of copying parts of the texts from another poem for *Lydgate's Kings of England* in the same period [29]. In phylogenetics, lateral gene transfer is a process in which an organism incorporates the genetic material from another organism, which is not its ancestor in the evolutionary history.

Third, convergent evolution also exists in stemmatology. The same change can be

introduced independently rather than by the propagation process. For example, people in the same geographical area might make the same modification of a word as a result of their common dialect [29]. The coincidence of changes makes unrelated lineages of documents more similar and thus more likely to be grouped together. In sum, similar to their parallels in phylogenetic analysis, the processes discussed above make the finding of the true tree in stemmatology more complex.

## 4.2 Computer-assisted methods in stemmatology

This section discusses how to apply computational methods in stemmatic analysis. First, we discuss the preprocessing steps of how to align texts and how to code the aligned texts to the format that is required by most programs. After that, we introduce how to evaluate a stemmatic tree. Finally we review several computational methods that are applied in stemmatic analysis.

### 4.2.1 Aligning and encoding text data

Like in phylogenetic analysis, after the raw data of a set of documents is provided, the first step is to align the documents to a matrix with each column containing a set of homologous words. But unlike the sequence data in phylogenetics, the stemmatic documents are less likely to have repeated regions. A word usually appears only several times in different sites of the documents. Therefore, a sub-optimal alignment is less probable in the text alignment, which makes the multiple text alignment comparably easier than the multiple alignment of genetic sequences. We discuss a multiple text alignment program based on the global alignment algorithm in the appendix of the thesis.

After the alignment, the texts need to be coded to the format that is applicable in the tree reconstruction programs. For example, the widely used NEXUS format [39] can be used for coding text data. To code the aligned text matrix to the NEXUS format, each column is considered as an independent group. Different words in this group are simply assigned with different characters. Usually the characters which are used for coding the amino acids are used for coding the aligned texts because the number of characters available is much more than that are used for coding the nucleotides. But they still have a limit of no more than 21 different words for each column. This coding method is simple and can be done without human intervention. A more complex coding scheme is to code words based on their relationships [29].

First, a base word, for example the most common word, is determined for each column. Other words that are different from this base word are coded based on how they can be derived from the base word. Different characters are assigned to different situations, including: different word that changes the meaning, different word that does not change the meaning, word that affects the rhyme, words that change their order, *etc.*. The coding scheme aims to reveal the evolutionary relationships of the words in each column. It may serve as a starting point for further developing the evolution models for the stemmatic tree reconstructions based on these relationships. This coding scheme requires expert knowledges to determine the relationships of words. For example, whether the meaning of a word is changed or not can not be determined by computational methods. When the data set grows larger, it becomes a very laborious work to code all the words by hand. In this thesis, the coding of the aligned texts adapts the first way, which encodes words without considering their relationships.

### 4.2.2 Evaluation of the results in stemmatology

When the true structure is known, the average sign similarity method measuring the distance of two networks can be used to evaluate the accuracy of the resulting stemmatic tree. Unlike early distance measuring methods that are restricted to the bifurcating tree structures, this method can compute distances between arbitrary network structures [51]. The distance of two nodes inside a graph is the shortest path between them. Denote the distance of node $i$ and node $j$ as $d(i,j)$ in the first graph and $d'(i,j)$ in the second graph, where the sign of $x$ is represented as $sign(x)$. The index $\iota$ of the triplet $\{i,j,k\}$ in two graphs can be calculated as:

$$\iota(i,j,k) = 1 - \frac{1}{2}|sign(d(i,j) - d(i,k)) - sign(d'(i,j) - d'(i,k))|. \tag{74}$$

The index $\iota(i,j,k)$ equals 1 when $i$ is closer, more distant, or the same distance to $j$ than to $k$ in both graphs. When $i$ is closer to one of $j$ and $k$ in the first graph, but is closer to the other node in the second graph, $\iota(i,j,k)$ equals 0. Finally, $\iota(i,j,k)$ equals 0.5 when in one graph $i$ is in the middle of $j$ and $k$, but in the other graph, $i$ is closer to one of the nodes. The average sign similarity is then calculated by taking the average of $\iota$ for all observed triplets in the graphs. The hidden nodes are omitted in the calculation of $\iota$, which implies that two graphs with the same set of observed nodes are comparable even when they have different number of hidden nodes. In this thesis, this method is used for accessing the accuracies of the stemmatic and phylogenetic trees reconstructed by computational methods.

### 4.2.3 Computational methods in stemmatology

As discussed in Section 4.1, the evolution of manuscripts is analogous to the sequence evolution in biology in many aspects. Therefore, phylogenetics programs can be applied in computer-assisted stemmatology. In the thesis, the methods including the neighbor-joining (NJ) method [54], the least-squares (LS) method [13], the parsimony method [12], the maximum likelihood (ML) method using heuristic search algorithms [9], and the maximum a posteriori method using simulation algorithms [50] are compared to the structural EM method in stemmatic analysis. See Section 3.4.1 for the details of these methods and the phylogenetic programs which use them. In addition, the RHM method developed by Roos *et al.* [52, 51] is also compared with the structural EM method. The RHM method tries to find a stemma which can be coded with the shortest code length. In the learning process, the tree structure and unknown manusripts in the internal nodes are both updated by an algorithm combining stochastic optimization and dynamic programing. The RHM method was proved to be generally good in the *Parzival*, *Notre besoin*, and *Heinrichi* data. For the most complex data set, the *Heinrichi* data set, it had the best score in the experiment carried out by Roos *et al.* [51].

## 4.3 Structural EM for stemmatic tree reconstruction

This section illustrates the algorithm and models for the structural EM method in stemmatology.

### 4.3.1 Algorithm

The tree reconstruction in stemmatology aims to solve the evolutionary relationships of a set of documents. This process is similar to phylogenetic analysis for reconstructing relationships of different species. In the stemmatic tree reconstruction, the documents are aligned to a matrix with each row containing one document, each column containing the words which are genealogically related. The evolution of words in each column can also be assumed as a CTMC process with the properties of reversibility and lack of memory. However, there are two differences in the stemmatic tree reconstruction compared to the phylogenetic tree reconstruction. First, the stemmatic tree is not constrained to be a binary tree. Thus, in the structural EM method for the stemmatic tree reconstruction, no transformation procedure is

needed after learning the maximum spanning tree. Second, in phylogenetic analysis, the same molecular evolution model can be used in different positions of the aligned sequences. To analyze DNA sequences, a $4\times4$ matrix containing the transition probabilities between the four possible nucleotides is needed. For the protein sequences analysis, we need a $20 \times 20$ matrix for the amino acid evolution. But in stemmatology tree analysis, there are much more different elements in a data set. Therefore it is impractical to formulate a common transition matrix for the whole data set. However, this can be solved by modeling each column independently. In this case, each column has a transition probability matrix which is formulated only for the different words in that column. Still, this model brings another problem: the dependence of the transition probabilities on time $t$ are different in different columns. Therefore, when calculating the expected Bayesian score in the structural EM method, it becomes complicated to optimize the weight which is the sum of different functions of $t$. In this thesis, the transition probability in stemmatology is assumed to be fixed and have no dependence on the time. Thus, the structural EM method for stemmatology analysis can be performed as:

1: Initial a stemmatology tree by the neighbor-joining method.
2: **repeat**
3: For all node pairs, compute the expected Bayesian score

$$\begin{aligned} Q(\mathbf{T} : \mathbf{T}_i) &= \sum_{(i,j)\in\mathbf{T}} \sum_{(a,b)\in\Sigma^2} E[N(x_i = a, x_j = b)(\log P_{a\to b} - \log P_b)|\mathbf{T}_l] + C \\ &= \sum_{(i,j)\in\mathbf{T}} E[w_{i,j}(t_{i,j})] + C \end{aligned} . \tag{75}$$

The score is calculated based on topology $\mathbf{T}_l$ inferred in the previous step in the way described in Section 3.3.2. Perturb edge weights or position weights by the simulated annealing method as described on Section 3.3.5. {E-step}
4: Updating the previous topology $\mathbf{T}_l$ to current $\mathbf{T}_{l+1}$ using maximum spanning tree algorithm. {M-step}
5: **until** The expected Bayesian score converged, or run out of time.

### 4.3.2 Models

To calculate the probability of the changing of words, three models are utilized. The first model is named as the uniform model. It sets the probabilities of any substitutions between words to be the same. The probability of a word staying

unchanged is 0.95. Thus, for each column of the aligned texts, the probability of changing from one word to some other word is calculated by dividing 0.05 by the number of other unique words. To accomplish the reversibility in the evolution process, the probabilities of the occurrences of all the words in each column are assumed to be the same. If in one column there are $\kappa$ different words, and $a$ and $b$ are two of the different words in that column, the model can be represented as

$$\begin{cases} P_a = 1/\kappa \ , \\ P_{a\to a} = 0.95, \text{ and} \\ P_{a\to b} = 0.05/(\kappa - 1). \end{cases} \tag{76}$$

The second model sets the probability of a word changing to another word to be proportional to the probability of the occurrence of the target word in the data set. At the same time, the probability of the occurrence of a word is calculated by counting the occurrences of that word. This model is similar to the F81 model in phylogenetics except it calculates the probabilities of occurrences and transitions of words in each column of the aligned texts separately. Denote the number of the occurrence of a word $a$ in a column as $N(a)$, the F81-like model is

$$\begin{cases} P_a = N(a)/\sum_{b\in\Sigma} N(b), \\ P_{a\to a} = 1 - \sum_{b\in\Sigma, b\neq a} P_{a\to b}, \text{ and} \\ P_{a\to b} = 0.1 P_b. \end{cases} \tag{77}$$

In the third model, denoting the measure of the difference between two words as $\xi$, then the probability of changing between the two words is $1/2^{\xi}$. The probability of the occurrence of a word is then calculated based on the property of the reversibility in the evolution of words. The difference between two words is measured by the least information, or the code length, needed for changing between the two words. It is an instance of the minimum description length [49]. Therefore, this model is named as the MDL model. In practice, the information can be represented by the Kolmogorov complexity [37] and approximated by the compression program *gzip*. For instance, the amount of information needed for $a$ changing to $b$ is calculated by subtracting the bits needed for compressing word $a$ alone, from bits needed for compressing $a$ concatenated with $b$.

Here is an example of how to calculate the transition probabilities under the three models. In the artificially generated *Heinrichi* data [51], one of the columns contain

Table 2: The transition probabilities of *Angliassa* to other words and itself using different models.

| Target word | Uniform model | F81-like model | MDL model |
|---|---|---|---|
| *Angliassa* | 0.950 | 0.904 | 0.443 |
| *angliassa* | 0.010 | 0.004 | 0.491 |
| *Caalimaassa* | 0.010 | 0.018 | 0.023 |
| *Caalimaasta* | 0.010 | 0.004 | 0.005 |
| *caalimaassa* | 0.010 | 0.068 | 0.023 |
| *taalainmaassa* | 0.010 | 0.004 | 0.062 |

six different words: *Angliassa*, *angliassa*, *Caalimaassa*, *Caalimaasta*, *caalimaassa*, and *taalainmaassa*, with the probabilities of the occurrences of the words as $\{0.036, 0.036, 0.179, 0.036, 0.679, 0.036\}$. We show the transition probabilities of *Angliassa* to other words and itself using different models in Table 2. Using the uniform model, the transition probabilities of *Angliassa* to other words are all the same. On the other hand, in the F81-like model, they are proportional to the target words. The transition probability of *Angliassa* changing to *Caalimaassa* is higher than the probability of *Angliassa* changing to *angliassa*, because *Caalimaassa* has a higher probability of occurrence than *angliassa*. But using the MDL model, the transition probability of *Angliassa* changing to *Caalimaassa* is lower than to *angliassa*. The reason is that *Angliassa* is more similar to *Angliassa* than *Caalimaassa*. Thus the bits needed for *Angliassa* changing to *Caalimaassa* is less than the bits needed for *Angliassa* changing to *angliassa*.

## 4.4 Experiments

In the following sections, we first discuss how to generate the artificial stemmatic data. Then, we present the results of three data sets separately. In the end, we give a generalization of the performance of the structural EM method for learning the stemmatic data.

In the initial steps of the structural EM method for these three data sets, when we had $N$ texts in the data set, we added $N - 2$ of hidden nodes. At the end, some hidden nodes were put in the outer layers of the learned trees. These hidden nodes are not shown in the resulting stemmatic trees. On the other hand, the internal hidden nodes are represented as points in the resulting trees. Moreover, the

stemmatic trees are unrooted. They are exhibited in the orders which are easy to compare with the correct structures.

### 4.4.1 Artificial stemmatic data

To evaluating the performance of the structural EM method, three artificially generated data sets were used, including *Notre besoin* [1], *Parzival* [59], and *Heinrichi* [51]. The three data sets were generated in similar ways. The scribes were asked to copy spontaneously without any advice of how to correct the texts. All the artificial traditions used unfamiliar contents, with a relatively undemanding level of languages to the scribes. *Notre besoin* contained texts from a literary prose, *Notre besoin de consolation est impossible à rassasier*, which was translated to French [51]. All the scribes except one for the *Notre besoin* data set were French-speaking. *Parzival* data used a medieval German poem which was translated to English for the English speaking scribes. The texts in *Heinrichi* data were written in old Finnish and copied by Finnish-speaking scribes. The unfamiliar language for the scribes in creating artificial text traditions resembles the situation what the medieval scribes had when copying the Latin vernacular texts, which were no longer spoken or written as the mother tongue in the Middle Age. In addition, the partially recognizable language ensured that enough mistakes were made during the copying process.

In the copying process of the three artificial data sets, some scribes copied the texts for more than once. As a result, they might be influenced by the previous knowledge of the contents of texts when they copied the following versions. This situation simulated the copying of the liturgical and biblical texts in Middle Age when the contents of texts were usually familiar to the scribes.

The complexities of the three data set are roughly in the ascending order of *Parzival*, *Notre besoin*, and *Heinrichi*. To make the artificial data more realistic, some copies in the data sets were held back. In the *Heinrichi* data, approximately half of the created manuscripts were hidden from the data set. It was to simulate the case when a significant portion of manuscripts were lost in the history, thus unavailable for modern researchers. Additionally, the situation of contamination, where more than one exemplars are used as sources also exists in *Notre besoin* and *Heinrichi*. 8% of the textual witnesses of *Notre besoin*, and 12% of *Heinrichi* contained contaminations. Besides, in the *Heinrichi* data, some manuscripts had parts of their contents deleted to simulate the case that only parts of the manusript was recognizable. Furthermore, the three data sets have different numbers of parsimony informative positions, which

Table 3: Summary of the artificially generated stemmatic data sets.

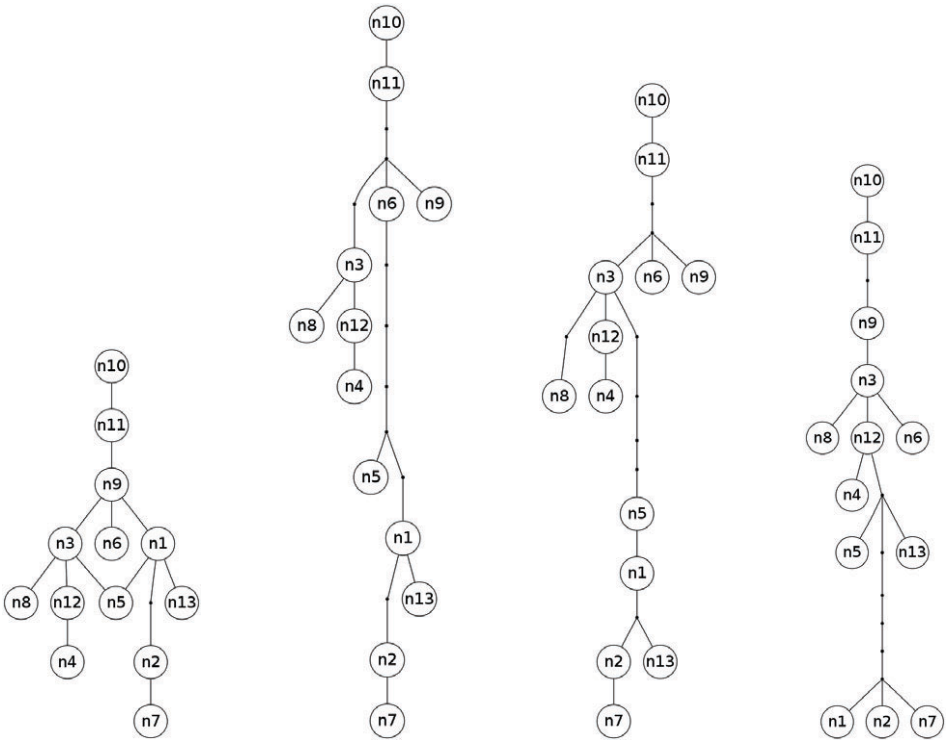
| Data | Length[a] | PIP [b] | MS [c] | Missing [d] | Contamination [e] |
|---|---|---|---|---|---|
| Notre besoin | 1,035 | 71 | 14 | 1(7%) | 1(7%) |
| Parzival | 855 | 88 | 21 | 5(24%) | 0(0%) |
| Heinrichi | 1,208 | 617 | 63 | 30(45%) | 4(12%) |

[a]Maximum length of the documents in words
[b]Parsimony informative positions
[c]Number of the manuscripts
[d]Number of the manuscripts missing from the data
[e]Number of the manuscripts contaminated

also indicates the complexity of the data sets. The position which is not parsimony informative will not affect the number of substitutions needed, no matter which tree structure is chosen. A parsimony informative position is the position where there are no less than two variants which existe in at least two manuscripts. Among all the three data sets, the *Notre besoin* data has the least number of parsimony informative positions. The parsimony informative positions in *Heinrichi* data are about ten times of the other two data set. Table 3 gives a summary of the three data sets.

However, there are some important differences between the artificial and real manuscript traditions. First, the periods between the copies of artificial generated data are very small, usually no more than a few weeks. In the real tradition, years or centuries might exist between different copies. The historical element is one of the important influences in the manuscript evolution, but it can not be simulated in the artificial generated data. Besides, in the real tradition, different copies may spread in a large geographical area. Different languages used in different areas could have important effects on the manuscript evolution. However, there is no significant space effect in the artificial generated data.

### 4.4.2 *Notre besoin* data




(a) Correct structure of the *Notre besoin* data

(b) Stemmatic tree of the *Notre besoin* data learned by the SEM method using the uniform model

(c) Stemmatic tree of *Notre besoin* data learned by SEM method using the F81-like model

(d) Stemmatic tree of *Notre besoin* data learned by SEM method using the MDL model

Figure 11: Correct stemmatic tree and stemmatic trees learned by the structural EM method using the uniform, F81-like and MDL models for the *Notre besoin* data. The trees learned by the structural EM method are ordered manually to be easy to compare to the correct tree.

The *Notre besoin* data set is a relatively simple data set with only thirteen documents. Besides the structural EM method, all other methods got good results. The neighbor-joining method with bootstrapping and the RHM method performed best with the average sign similarities of about 77%. Other Bayesian based methods including the simulation method of MrBayes and the maximum likelihood method using the heuristic search algorithms in PHYLIP also obtained satisfactory results.

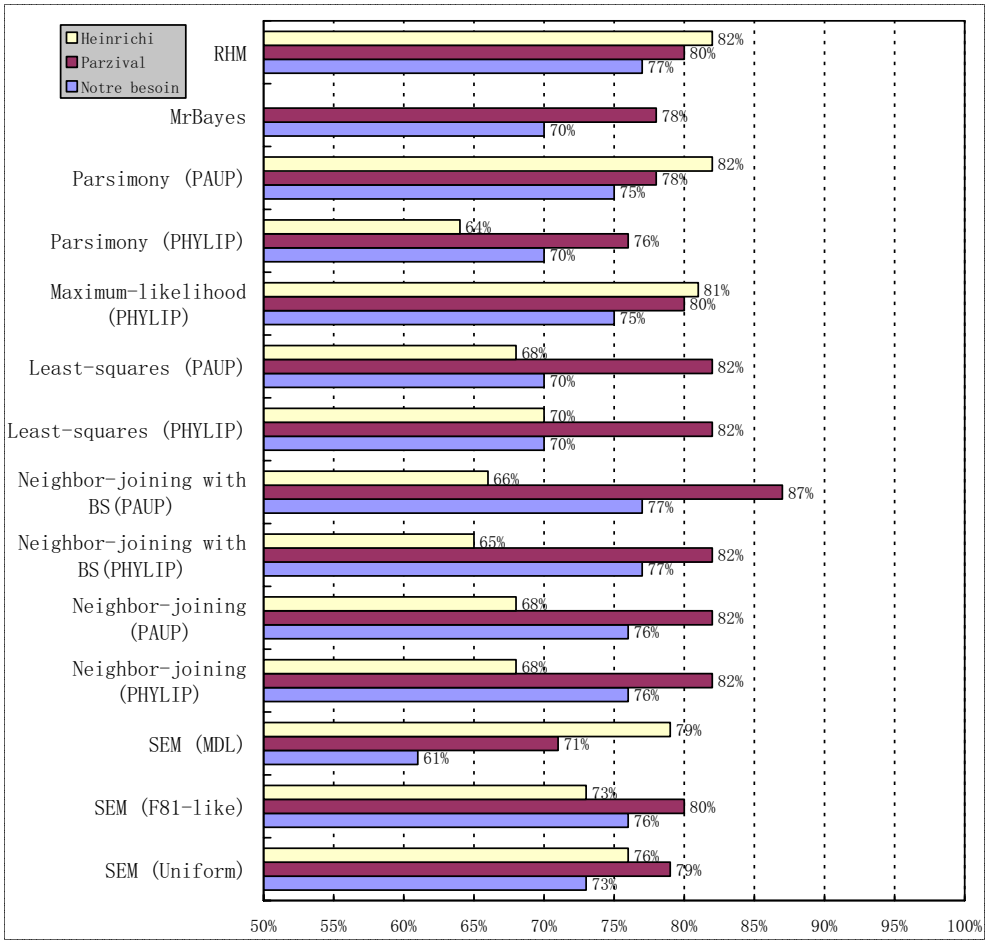



Figure 12: Average sign similarities of the stemmatic trees learned by different methods on the artificial generated stemmatic data sets of *Notre besoin*, *Parzival* and *Heinrichi*. The average sign similarities are shown from 50% in the bar chart.

The structural EM method using the three models resulted in different accuracies. The average sign similarity of the structural EM method using the MDL model was 15% (units) less than the structural EM method using the F81-like model. This suggested that the model selection in structural EM method was very important for learning a good structure for the stemmatic data. Table 4 and Figure 12 shows the average sign similarities of different methods on *Notre besoin* and other data sets.

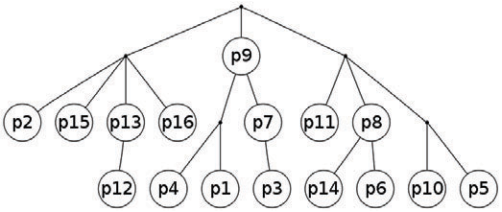
All the models found three groups of nodes: {n6, n9, n10, n11}, {n3, n8, n12, n14}, and {n1, n5, n13, n2, n7}. Furthermore, the structures of {n3, n8, n12, n14} learned by the structural EM method using the uniform model and the MDL model

are exactly the same as the correct structure. Besides, the structural EM method could identify the documents which were not in the leafs of the stemmatic trees. For example, the internal documents: n1, n2, n3, n11, and n12, were in the middle positions of the stemmatic trees in the results by the uniform and F81-like models. On the other hand, other programs programs could not recognize the internal nodes, which could be the ancestors of some other documents. However, none of the methods including the structural EM method, are able to identify the contaminations in the data set. There is one contamination in this data set. Document n5 was copied from both n1 and n3. This might cause the structural EM method to recognize n1 as the child of n5, and then link the group {n1, n5, n13, n2, n7} to other groups by n5. In the correct structure, the group {n1, n5, n13, n2, n7} is linked to other groups by both n1 and n5.

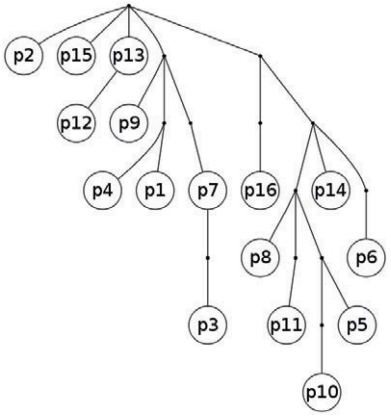
Table 4: Average sign similarities of the stemmatic trees learned by different methods on artificial generated stemmatic data sets of the *Notre besoin*, *Parzival* and *Heinrichi* data sets.

| Method | *Notre besoin* | *Parzival* | *Heinrichi* |
|---|---|---|---|
| Structural EM using uniform model | 73% | 79% | 76% |
| Structural EM using F81-like model | 76% | 80% | 73% |
| Structural EM using MDL model | 61% | 71% | 79% |
| Neighbor-joining (PHYLIP) | 76% | 82% | 68% |
| Neighbor-joining (PAUP) | 76% | 82% | 68% |
| Neighbor-joining with BS (PHYLIP) | **77%** | 82% | 65% |
| Neighbor-joining with BS (PAUP) | **77%** | **87%** | 66% |
| Least-squares (PHYLIP) | 70% | 82% | 70% |
| least-squares (PAUP) | 70% | 82% | 68% |
| Maximum-likelihood (PHYLIP) | 75% | 80% | 81% |
| Parsimony (PHYLIP) | 70% | 76% | 64% |
| Parsimony (PAUP) | 75% | 78% | **82%** |
| MrBayes | 70% | 78% | - |
| RHM | **77%** | 80% | **82%** |

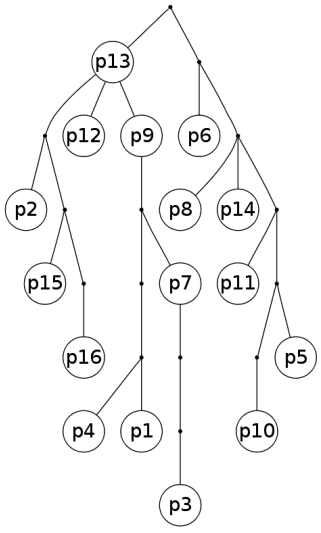
### 4.4.3 *Parzival* data



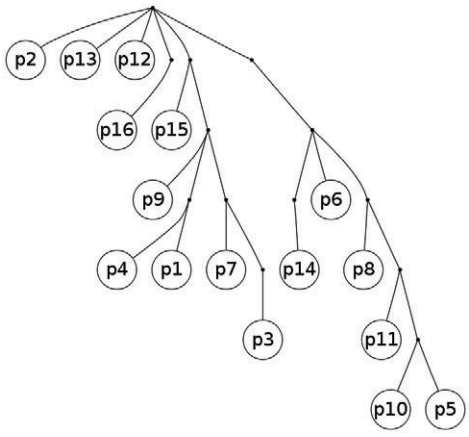

(a) Correct structure of the *Parzival* data




(b) Stemmatic tree of the *Parzival* data learned by the SEM method using the uniform model




(c) Stemmatic tree of the *Parzival* data learned by the SEM method using the F81-like model




(d) Stemmatic tree of the *Parzival* data learned by the SEM method using the MDL model

Figure 13: The correct stemmatic tree and stemmatic trees learned by the SEM method using the uniform, F81-like and MDL models for the *Parzival* data. The trees learned by the structural EM method are ordered manually to be easy to compare to the correct tree.

The *Parzival* data has sixteen documents and no contamination. All methods obtained good results. The neighbor-joining method with bootstrapping in PAUP

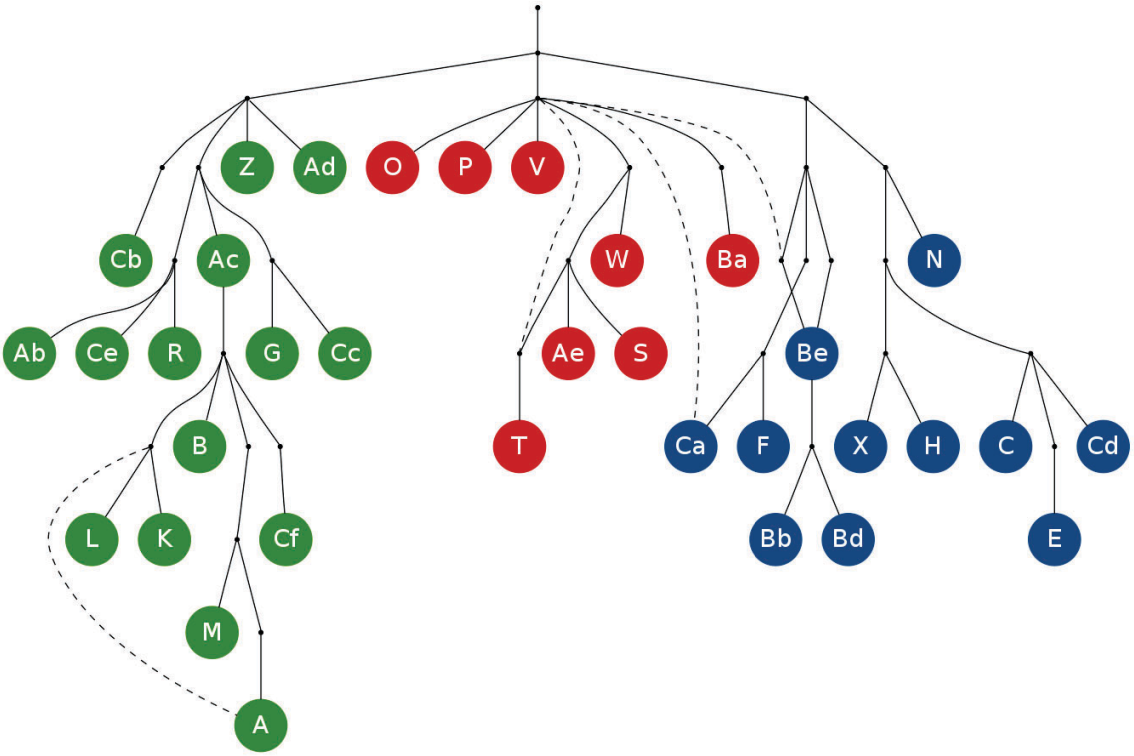
performed the best with average sign the similarities of 87%. The results by the structural EM method with different models achieved the average sign similarities of no less than 71%.

The structural EM method with different models could find most of the node groups. For example, the F81-like and MDL models correctly found three groups of nodes, which were {p2, p15, p13, p12, p16}, {p1, p3, p4, p7, p9}, and {p5, p6, p8, p10, p11, p14}. For the two groups: {p2, p15, p13, p12, p16} and {p1, p3, p4, p7, p9}, the structural EM method using the uniform model and the F81-like model also obtained similar structures as the original structures except the number of hidden nodes located within these groups.

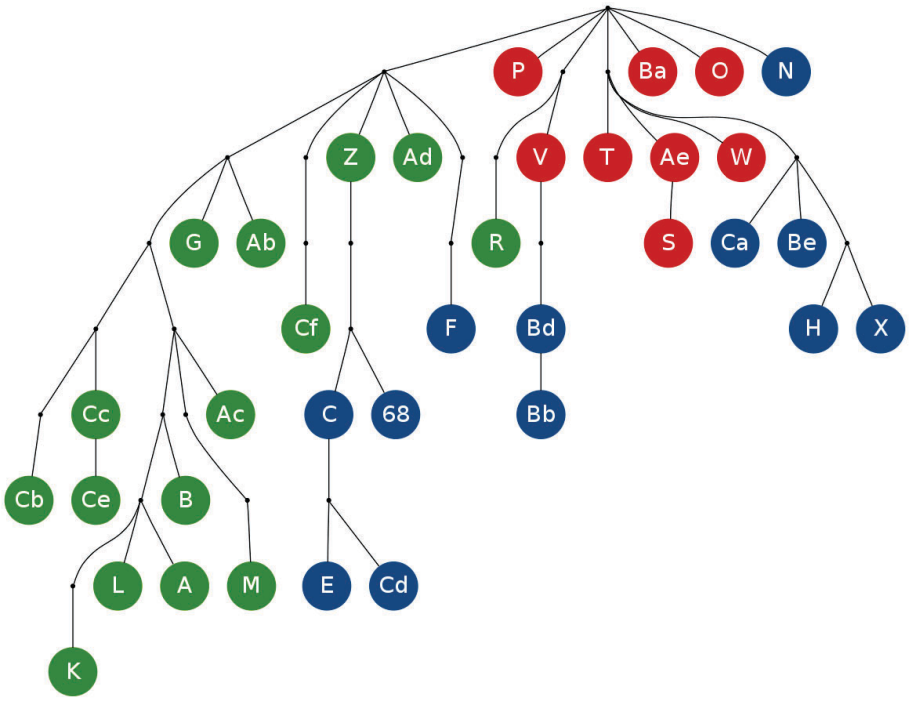
Furthermore, The structural EM method with different models could find most of the internal documents. However, p8 was not recognized in any of the resulting trees. In the correct structure, p8 is the parent of p14 and p6, and is closer to p9 than its children. In all results by the structural EM method using different models, p8 was located equal or more distant from p9 than p14 and p6. To explain this, the distance method was applied, which measured the distance of two documents by how many modifications were needed to change from one document to the other. It showed that p9 was closer to p14 and p6 than p8. This might be one of the reasons for structural EM method to put p14 and p6 in the middle of p8 and p9, thus unable to identify p8 as the parent of p14 and p9.

### 4.4.4 *Heinrichi* data

The *Heinrichi* data set is more complex with a large portion of missing data and four cases of contaminations. In Figure 14, the nodes are colored in three groups for easy recognition. The contaminated documents are linked with extra sources with dashed lines. The structural EM method with three models had good results. However, some other methods, for instance the neighbor-joining and least-squares methods, which were relatively good for smaller data sets failed to give satisfactory results for the *Heinrichi* data set. The parsimony method in PAUP kept performing well. On the other hand, the parsimony method in PHYLIP produced the structure with the smallest average sign similarity. This difference might due to the different search algorithms utilized by the two programs, but not the method itself. The RHM method also performed satisfactory. The RHM method and the parsimony method in PAUP achieved the best average sign similarities of 82%. The maximum likelihood method in PHYLIP using the heuristic search algorithm obtained second

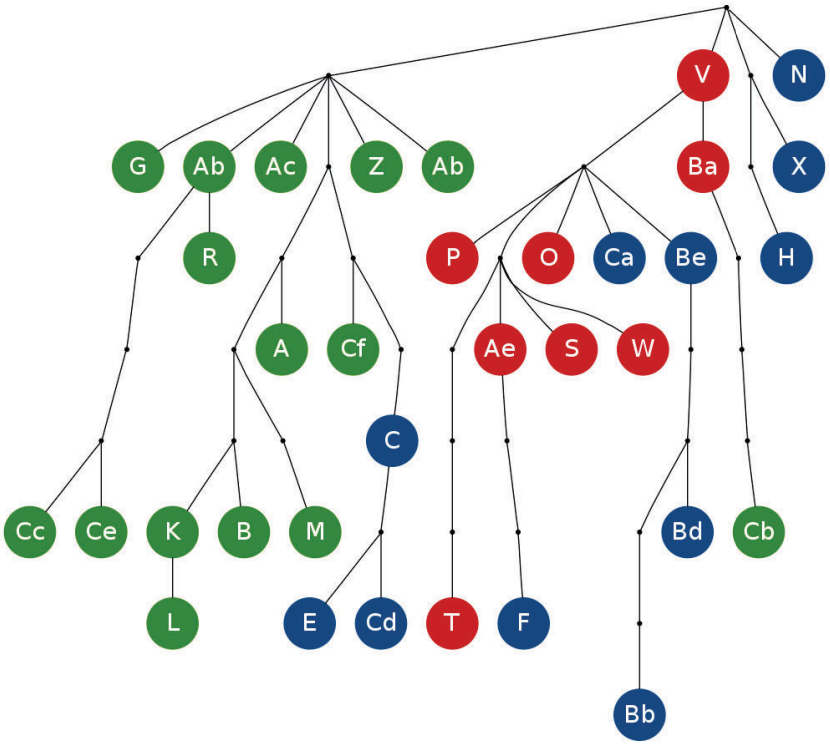



(a) Correct structure of the *Heinrichi* data

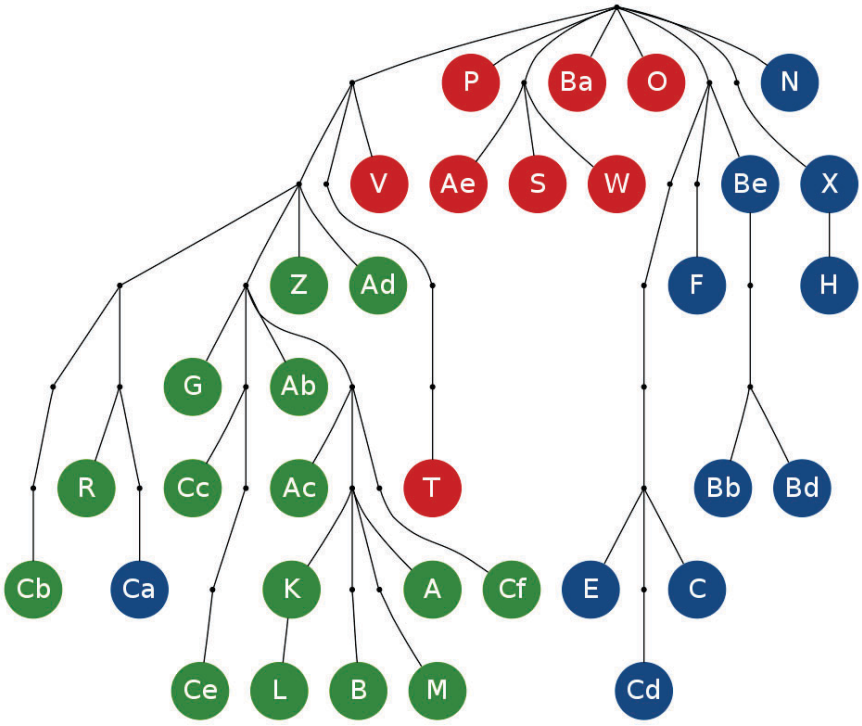



(b) Stemmatic tree of the *Heinrichi* data learned by the SEM method using the uniform model

Figure 14: Original tree and trees learned by the SEM method using the uniform for the *Heinrichi* data. The trees learned by structural EM method are rooted manually to be easy to compare to the correct tree.




(c) Stemmatic tree of the *Heinrichi* data learned by the SEM method using the F81-like model




(d) Stemmatic tree of the *Heinrichi* data learned by the SEM method using the MDL model

Figure 14: The trees learned by the SEM method using the F81-like and MDL models for the *Heinrichi* data. The trees learned by structural EM method are rooted manually to be easy to compare to the correct tree.

best score of 81%.

The structural EM method with the MDL model produced third best score, 79%. The resulting stemmatic trees learned by the structural EM method with the F81-like obtained the average sign similarities of 6% (units) less than the stemmatic tree learned by the structural EM method using MDL model. Generally, the stemmatic tree by the MDL model could separate the three colored groups. However, blue group node Ca was located inside the green group nodes. The resulting trees by the uniform model and the F81-like model could only cluster most of the green group nodes together. The red and blue group nodes were mixed. It might partly due to Ca and Be in blue group, which have sources both from the red and blue group nodes. Therefore, Ca, Be and their descendants could be clustered in the red group too.

In the three resulting trees, the contaminated nodes were located differently. For example, originally, A has sources from both the parent of {K, L}, and the sibling of M. In the stemmatic trees learned by the uniform and MDL models, A was located closer to {K, L} than to M. However, In the stemmatic tree learned by the F81-like model, A was located equidistant to K and M. This suggests that there is no rules for locating the contaminated nodes. It can be located closer to any of its sources in the stemmatic trees learned by the structural EM method.

For the smaller groups of documents including: {Ae, S, T}, {O, P, V}, {Ca, F}, {Be, Bb, Bb}, {C, Cd, E}, {H, X}, {G, Cc}, {Z, Ad}, {Ab, Ce, R}, and {K, L}, the three models also had different performances in recognizing them. The stemmatic trees using different models had some common problems to recognize some groups, such as {Ca, F} and {Ab, Ce, R}. But they could also have model specific mistakes. For example, only the result using the uniform model was not able to identify group {Be, Bb, Bb}. This further proved the importance of the model selection in learning the true stemmatic tree.

In the cases of other Bayesian-based methods, the *Heinrichi* data was too complex for the simulation method in MrBayes to converge and build a stemmatic tree. On the other hand, the maximum likelihood method using the heuristic search algorithm in PHYLIP had the average sign similarity of 81%. This result was only 2% (units) more than the score of the structural EM method using the MDL model. It suggested that the structural EM method was more suitable for finding the maximum likelihood tree for the large data set than the simulation method, and at least as good as the Bayesian method using the heuristic search algorithms. The

distance-based method of the neighbor-joining and least-squares methods failed to find reasonable stemmatic trees for the *Heinrichi* data. This may due to the increasing complexity of the data: the longer texts , the greater parts of the missing data and more cases of contaminations. The parsimony method in PAUP still performed well but the same method in PHYLIP generated a bad result. It indicated that the specific algorithms were the reasons which caused the different performances of the parsimony method in the different programs. The parsimony method was suitable for complex data if a proper algorithm was utilized.

# 5 Discussion

In phylogenetic analysis, the structural EM method obtained consistent results for all the artificially data sets. However, it could not find a structure as good as the parsimony method and the maximum likelihood method using heuristic algorithms. These two methods usually obtained the best results (more than 90% of the average sign similarities) for all the data sets. The reason might be the structural EM method's tendency to be stuck in local maxima. In the analyzing of the regulatory genes from *streptococcus pneumoniae*, the resulting phylogenetic trees by the structural EM method are usually biological consistent, and had similar structures as the results by Varvio *et al.*.

The structural EM method also performed generally well for stemmatic data. It could find most node groups correctly in a large data set such as the *Heinrichi* data. But the exact locating of groups inside a tree was a much harder task for the structural EM method. It usually could not correctly locate all the groups. The structural EM method does not restrict the structure to be a bifurcating tree with the observed data only in leaf nodes. This feature makes it superior in learning stemmatic data which contains ancestral documents in the internal positions of the stemmatic tree. For example, in the *Notre besoin* data set, the structure of node group {n3, n8, n12, n14}, with n3 and n12 as internal nodes, in the resulting trees using the uniform and MDL models are exactly the same as the original structure. It is impossible for other programs to learn this kind of structures because they allow no internal nodes for the observed data. In addition, the bifurcating tree assumption makes the resulting stemma more complex by requiring a fixed number of the internal nodes, which can be larger than in the real case. Thus, the documents that are close to each other in the correct stemma are forced apart by adding more

hidden nodes between them. However, in the learning process of the structural EM method, the hidden nodes can be put in the leaf positions. Thus, the chances of adding redundant nodes between documents are reduced.

Two methods, the RHM method and the Parsimony method from the PAUP program also performed consistently for all the data sets. The parsimony method from PHYLIP had much worse performance than the parsimony method from PAUP in the *Heinrichi* and *Notre besoin* data sets. It suggested that different algorithms in the frame of the same method had an important effect in the accuracies of the results. The maximum likelihood method in PHYLIP using the heuristic search algorithm also performed consistently well for all the data sets. On the other hand, MrBayes failed to generate the trees for the large data set, the *Heinrichi* data set. The neighbor-joining method and the least-squares method were suitable for treating small data sets but performed bad in the larger data set. However, the more complex data set can be more realistic than the smaller one. Therefore, the parsimony method, the RHM method, and the structural EM method can be more suitable tools for analyzing the real data.

In the future, there are several aspects that need to be developed for the structural EM method for phylogenetic and stemmatic analyses. First, we need to increase the accuracy of the structural EM in phylogenetic analysis, possibly by adapting more powerful ways to deal with the problem of local maxima. Secondly, the structural EM method could not recognize the situation of contaminations. The node that has multiple sources was usually located close to one of its source in the stemmatic tree by the structural EM method. In addition, the contaminations always caused the structural EM method to make mistake for locating node groups which contained contaminated nodes. Thirdly, the edge lengths were not learned in the structural EM method for stemmatic analyses. The edge lengths can suggest some important information in stemmatic analysis, such as how different two versions of documents are. They may have important effects in the resulting stemmatic tree. For example, a long edge between two nodes can replace several internal nodes between them. Thus, the number of the internal nodes needed can be reduced if the edge lengths are learned. A further exploration of the models in stemmatology is needed to incorporate the learning of the edge lengths in the structural EM method. Finally, the learned trees were not rooted in the structural EM method or other methods. In the phylogenetic study, usually an outgroup species can be used for locating the root of a phylogenetic tree. The outgroup species is the oldest species which is very distantly from the other species in the phylogenetic tree. It is recognized as the

root of the phylogenetic tree. The direction of time is from this outgroup species. However, in stemmatic analysis, the information of outgroup is usually not available. Thus, some other ways for identifying the root, possibly along the lines of [6], need to be considered and explored.

# References


1 P. V. Baret, C. Macé, and P. Robinson. Testing Methods on an Artificially Created Textual Tradition. In *Linguistica computazionale*, pages 255–281. Istituti Editoriali e Poligrafici Internazionali, 2006.

2 D. Barker. *LVB 1.0: Reconstructing Evolution with Parsimony and Simulated Annealing.* University of Edinburgh, Edinburgh, 1997.

3 I. A. Beinlich, H. J. Suermondt, R. M. Chavez, and G. F. Cooper. The ALARM Monitoring System: A Case Study with Two Probabilistic Inference Techniques for Belief Networks. *2nd European Conference on Artificial Intelligence in Medicine*, 38:247–256, 1989.

4 M. P. Berger and P. J. Munson. A novel randomized iterative strategy for aligning multiple protein sequences. *Computer Applications in the Biosciences*, 7(4):479–484, 1991.

5 J. Binder, D. Koller, S. Russell, and Kanazawa K. Local learning in probabilistic networks with hidden variables. *International Joint Conference on Artificial Intelligence*, 14(2):1146–1152, 1995.

6 J. L. Cisne, R. M. Ziomkowski, and S. J. Schwager. Mathematical Philology: Entropy Information in Refining Classical Texts' Reconstruction, and Early Philologists' Anticipation of Information Theory. *PLoS ONE*, 5:e8661, 1999.

7 J. J. Doyle. Gene Trees and Species Trees: Molecular Systematics as One-Character Taxonomy. *Systematic Botany*, 17(1):144–163, 1992.

8 A. Dress and M. Kruger. Parsimonious phylogenetic trees in metric spaces and simulated annealing. *Advances in Applied Mathematics*, 8:8–37, 1987.

9 J. Felsenstein. Evolutionary trees from DNA sequences: A maximum likelihood approach. *Journal of Molecular Evolution*, 17(6):368–376, 1981.

10 J. Felsenstein. Phylogenies from Molecular Sequences: Inference and Reliability. *Annual Review of Genetics*, 22:521–565, 1988.

11 J. Felsenstein. PHYLIP-Phylogeny Inference Package (version 3.2). *Cladistics*, 5:164–166, 1989.

12 W. M. Fitch. Toward defining the course of evolution: minimum change for a specific tree. *Systematic Zoology*, 20:406–416, 1971.

13 W. M. Fitch and E. Margoliash. Construction of phylogenetic trees. *Science*, 155:279–84, 1967.

14 W. Fletcher and Z. Yang. INDELible: A Flexible Simulator of Biological Sequence Evolution. *Molecular Biology and Evolution*, 26(8):1879–1888, 2009.

15 D. Freedman. *Markov Chains.* Springer, New York, 1971.

16 N. Friedman. Learning Belief Networks in the Presence of Missing Values and Hidden Variables. *14th International Conference on Machine Learning*, pages 125–133, 1997.

17 N. Friedman. The Bayesian structural EM algorithm. *Proceedings of the 14th International Conference on Uncertainty in Artificial Intelligence*, 1998.

18 N. Friedman, M. Ninio, I. Pe'er, and T. Pupko. A Structural EM Algorithm for Phylogenetic Inference. *Journal of Computational Biology*, 9:331–354, 2002.

19 M. Frydenberg. The chain graph Markov property. *Scandinavian Journal of Statistics*, 17:333–353, 1990.

20 S. Geman, D. Geman, K. Abend, and T. J. Harley. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

21 F. C. Gergory and T. Dietterich. A Bayesian Method for the Induction of Probabilistic Networks from Data. *Machine Learning*, 9:309–347, 1992.

22 O. Gotoh. A weighting system and algorithm for aligning many phylogenetically related sequences. *Computer Applications in the Biosciences*, 11(5):534–551, 1995.

23 O. Gotoh. Multiple sequence alignment: algorithms and applications. *Advances in Biophysics*, 36:159–206, 1999.

24 V. Granville, M. Krivanek, and J. P. Rasson. Simulated annealing: A proof of convergence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(6):652–656, 1994.

25 S. Greenland and W. D. Finkle. A Critical Look at Methods for Handling Missing Covariates in Epidemiologic Regression Analyses. *American Journal of Epidemiology*, 142(12):1255–1264, 1995.

26 D. Heckerman. A Tutorial on Learning With Bayesian Networks. *NATO ASI series D. Behavioural and Social Sciences*, 89:301–354, 1998.

27 D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20(3):197–243, 1995.

28 S. Holmes. Bootstrapping Phylogenetic Trees: Theory and Methods. *Statistical Science*, 18:241–255, 2003.

29 C. J. Howe, A. C. Barbrook, M. Spencer, P. Robinson, B. Bordalejo, and L. R. Mooney. Manuscript evolution. *Endeavour*, 25(3):121–126, 2001.

30 D. H. Huson and D. Bryant. Application of phylogenetic networks in evolutionary studies. *Molecular Biology and Evolution*, 23(3):254–267, 2006.

31 K. K. Kidd and Sgaramella-Zonta L. A. Phylogenetic analysis: concepts and methods. *The American Journal of Human Genetics*, 23:235–252, 1971.

32 M. Kimura. A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. *Journal of Molecular Evolution*, 16(2):111–120, 1980.

33 H. Kishino and M. Hasegawa. Evaluation of the maximum likelihood estimate of the evolutionary tree topologies from DNA sequence data, and the branching order in hominoidea. *Journal of Molecular Evolution*, 29(2):170–179, 1989.

34 P. Knees, M. Schedl, and G. Widmer. Multiple lyrics alignment: Automatic retrieval of song lyrics. *Proceedings of 6th International Conference on Music Information Retrieval*, pages 564–569, 2005.

35 B. F. Koop, D. A. Tagle, M. Goodman, and J. L. Slightom. A molecular view of primate phylogeny and important systematic and evolutionary questions. *Molecular Biology and Evolution*, 6:580–612, 1989.

36 J. A. Lake. The order of sequence alignment can bias the selection of tree topology. *Molecular Biology and Evolution*, 8(3):378–385, 1991.

37 M. Li and P. M. B. Vitànyi. An Introduction to Kolmogorov Complexity and Its Applications, 2nd. ed. In *Statistical decision theory and related topics*. Springer-Verlag, New York, 1997.

38 F. Lutzoni, P. Wagner, V. Reeb, and S. Zoller. Integrating ambiguously aligned regions of dna sequences in phylogenetic analyses without violating positional homology. *Systematic Biology*, 49(4):628–651, 2000.

39 D. R. Maddison, D. L. Swofford, and W. P. Maddison. Nexus: An Extensible File Format for Systematic Information. *Systematic Biology*, 46(4):590–621, 1997.

40 B. Mau, M. A. Newton, and B. Larget. Bayesian Phylogenetic Inference via Markov Chain Monte Carlo Methods. *Biometrics*, 55:1–12, 1999.

41 G. J. McLachlan and T. Krishnan. *The EM Algorithm and Extensions*. Wiley, New York, 1997.

42 J. W. Myers, K. B. Laskey, and T. Levitt. Learning Bayesian networks from incomplete data with stochastic search algorithm. *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pages 476–485, 1999.

43 S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453, 1970.

44 J. Neyman. Molecular studies of evolution: a source of novel statistical problems. In S. Gupta and J. Yackel, editors, *DTRT*, pages 1–27. New York Academy Press, 1971.

45 C. Notredame, D. G. Higgins, and J. Heringa. T-coffee: a novel method for fast and accurate multiple sequence alignment. *Journal of Molecular Biology*, 302(1):205–217, 2000.

46 A. Papoulis. *Probability, Random Variables, and Stochastic Processes, 2nd edition*. McGraw-Hill Companies, New York, 1984.

47 M. N. Radford. Probabilistic Inference Using Markov Chain Monte Carlo Methods. 1993.

48 B. D. Redelings and M. A. Suchard. Joint Bayesian estimation of alignment and phylogeny. *Systematic Biology*, 54(3):401–418, 2005.

49 J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.

50 F. Ronquist and J. P. Huelsenbeck. MrBayes 3: Bayesian phylogenetic inference under mixed models. *Bioinformatics*, 19:1572–1574, 2003.

51 T. Roos and T. Heikkilä. Evaluating methods for computer-assisted stemmatology using artificial benchmark data sets. *Literary and Linguistic Computing*, 24(4):417–433, 2009.

52 T. Roos, T. Heikkilä, and P. Myllymäki. A Compression-Based Method for Stemmatic Analysis. *Frontiers in Artificial Intelligence and Applications*, 141:805–806, 2006.

53 A. Rzhetsky and M. Nei. Theoretical foundation of the minimum-evolution. *Molecular Biology and Evolution*, 10:1073–1095, 1993.

54 N. Saitou and M. Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4:406–425, 1987.

55 D. S. Schwarz. How fast is local search. *26th Annual Symposium on Foundations of Computer Science*, pages 39–42, 1985.

56 G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

57 H. Shimodaira and M. Hasegawa. Multiple Comparisons of Log-Likelihoods with Applications to Phylogenetic Inference. *Molecular biology and evolution*, 16(8):1114–1116, 1999.

58 J. B. Slowinski. Molecular Polytomies. *Molecular Phylogenetics and Evolution*, 19(1):114–120, 2001.

59 M. Spencer, E. A. Davidson, A. C. Barbrook, and C. J. Howe. Phylogenetics of artificial manuscripts. *Journal of Theoretical Biology*, 227(4):503–511, 2004.

60 M. Spencer and C. Howe. Article: Collating Texts Using Progressive Multiple Alignment. *Computers and the Humanities*, 38(3):253–270, 2004.

61 J. Suzuki. Learning Bayesian networks base on the MDL principle: an efficient algorithm using the branch and bound technique. *IEICE Transactions on Information and Systems*, 82(2):356–367, 1999.

62 D. L. Swofford and D. P. Begle. *PAUP: phylogenetic analysis using parsimony. Version 3.1. User's manual.* Smithsonian Institution, Laboratory of Molecular Systematics, Washington, 1993.

63 D. L. Swofford, G. J. Olsen, P. J. Waddell, and D. M. Hillis. *Chapter 11 in Molecular Systematics, 2nd edition.* Sinauer Associates, Inc., Sunderland, 1996.

64 J. D. Thompson, D. G. Higgins, and T. J. Gibson. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22(22):4673–4680, 1994.

65 J. D. Thompson, F. Plewniak, and O. Poch. A comprehensive comparison of multiple sequence alignment programs. *Nucleic Acids Research*, 27(13):2682–2690, 1999.

66 E. Ukkonen. Approximate string-matching with q-grams and maximal matches. *Theoretical Computer Science*, 92(1):191–211, 1992.

67 S. L. Varvio, K. Auranen, E. Arjas, and P. H. Mäkelä. Evolution of the capsular regulatory genes in Streptococcus pneumoniae. *The Journal of Infectious Diseases*, 200(7):1144–1151, 2009.

68 W. Wheeler. Optimization alignment: The end of multiple sequence alignment in phylogenetics? *Cladistics*, 12(1):1–10, 1996.

69 W. Wheeler. Fixed Character States and the Optimization of Molecular Sequence Data. *Cladistics*, 15(4):379–386, 1999.

# A Multiple alignment

In the appendix, we focus on how to get a reliable multiple sequence alignment in phylogenetics and stemmatology. First, we review several popular algorithms for multiple alignment. Then we discuss how to address the problem of ambiguously aligned regions. These methods are usually suitable for phylogenetic data, but not suitable for stemmatic data with more unique words. Finally, we represent a program for aligning stemmatic sequences.

The multiple alignment is a critical step in phylogenetics because tree reconstruction methods usually require the aligned molecules in each position to be evolutionarily equivalent. Similarly in stemmatology, we need the documents to be aligned to a matrix with each row containing one document, each column containing the words which are genealogically related. The text alignment can be done in a similar way as aligning phylogenetic sequences. The following sections discuss several important computational multiple alignment methods.

Theoretically, the multiple alignment can be done by dynamic programming which aligns all the sequences together by searching through a multidimensional matrix to find the optimal aligning path. It is obvious that the amount of computing time and memory requirements increases exponentially as the number of the sequences, which determines the dimensions of the matrix. In practice, heuristic approaches are usually used, which include progressive methods, iterative methods, *etc.*.

## A.1 Progressive methods

Progressive methods build the tree using stepwise assemblies of aligned sequences. First, a guide tree is constructed by the neighbor-joining method. The distance matrix needed for the neighbor-joining method is built by calculating the similarity scores of the pairwise global alignment of all the possible pairs of the sequences. A guide tree does not have the rigor of a phylogenetic tree, but it can be used for guiding the order of adding sequence to the alignment. The assembly of the multiple alignment is performed by traversing the tree from leaves to root. Each time, the closest sequence pair is aligned by certain global alignment algorithm, for example, the Needleman-Wunsch alignment algorithm [43]. After that, a consensus sequence is produced and is assigned as the parent of the aligned sequences. In this way, all the sequences are aligned when the alignment process reaches the root.

Clustal [64] is one of the most widely used progressive multiple alignment programs. Besides the basic procedures in the progressive method, Clustal introduces other important features to assure a high quality alignment. It automatically selects the most suitable substitution matrix based on different sequence similarities. Another feature is that Clustal has adjustable gap penalties for different sites. In general, less gaps and insertions are allowed in conserved regions than in fast evolving regions. Finally, Clustal decreases the effects of groups of similar sequences, in order to improve the reliability of the alignment of more divergent sequences. This is done by down-weighting the alignment score of the sequence by its branch length to the root, which is then normalized by the number of nodes needed to traverse to the root.

However, there are some drawbacks in Clustal. First, it uses the global alignment which does not allow long gaps. So it is not suitable for aligning highly divergent sequences. Second, once an error is introduced in early steps, it can not be corrected in the following steps and will be propagated to the final result. What makes it worse is that in initial steps, errors are more likely to occur because there is less information to ensure a good alignment. Therefore, the alignment quality strongly depends on the order of the sequence addition. Another progressive aligning program, T-Coffee [45] can alleviate these limitations and is also widely used nowadays.

T-Coffee first performs the alignments of sequence pairs both globally and locally. Then it uses the library of results to evaluate the consistencies of the aligned molecules and gaps. The evaluation results are used to generate the scores for further refining the pairwise alignments. In the following steps, which are the pairwise alignments using the global method, the gap penalty is set to be zero because the scores calculated in the previous step have already included the information about gaps and insertions. Furthermore, a third sequence can be used as a reference to refine the resulting pairwise alignment. The guide tree is built based on the refined pairwise alignment scheme and the following progressive alignment is also executed using this scheme. Because multiple sources of information are used in the pairwise alignments in T-Coffee, it can minimize the errors in the early stage. T-Coffee is suitable for processing divergent sequences. However a complex pairwise alignment strategy in this program makes it usually slower than Clustal. In the experiment of analyzing capsular regulatory genes in *streptococcus pneumoniae*, Culstal was used for aligning the gene sequences.

## A.2 Iterative methods

Iterative methods modify the alignment in a hill-climbing manner until no refinement can be made. They do not guarantee to find the true optimal alignment, but have proved to be powerful in practice.

The program PRRN [4] uses the iterative method for multiple alignment. It performs the alignment iteratively in two steps. In the first step, a guide tree is built based on the initial random alignment for calculating the weight score for each sequence [22]. The weight score is used for compensating the contribution of the densely populated group. It has a similar aim as the weight scheme in Clustal, but is more complex by considering all possible paths that go through each pair of nodes. Another improvement is that PRRN uses the affine gap penalty which assigns more weight for opening a gap than continuing gaps. The penalty scheme allows long gaps and the continuous deletions to be modeled as a single event. In the second step, an inner iteration is performed. Sequences are randomly partitioned into two groups. Then, pairwise alignment is performed between the profiles of the two groups with the positions of sequences within each group fixed. Because the grouping of the sequences is random at each iteration, it can avoid the dependency of the order of adding the sequences to the alignment in progressive alignment methods. It can also correct the errors in the early stages, which is another critical problem in progressive alignment methods. The inner iteration is repeated until no improvement can be made to the alignment score. After that, it returns to the first step and builds a new guide tree based on the resulting alignment, from which the new weights are derived for another round of alignment in the second step. The outer iteration is performed until the overall alignment score converges.

The comparison study [65] shows that iterative methods are successful in getting more accurate alignments if enough information is provided. But they are unstable when the sequences are biased. Another disadvantage is its heavy time penalty compared to other methods. In the study performed by Thompson *et al.* [65], PRRN required the CPU time almost 80 times of the time which was required by ClustalX for the same set of sequences in the same computer environment.

## A.3 Joint estimation of alignments and trees

In the analysis of highly divergent sequences such as introns and ribosomal RNA sequences, the problem of ambiguously aligned regions becomes quite acute because

the position homology is likely to be violated in these regions. To accommodate the ambiguously aligned regions, some researchers may remove the whole or parts of them, while some researchers may just keep them even if they can not ensure the alignment quality in these regions. In addition to the heuristic ways, more sophisticated methods are developed to preserve the information in the ambiguously aligned regions as well as to optimize the alignment. For example, a score scheme can be introduced for coding the ambiguously aligned regions [38]. Another way is to refine the alignment while reconstructing the phylogenetic tree.

One approach is based on the maximum parsimony framework. In 1996, Wheeler [68] developed a method that searches through all possible topologies to find the one that needs the least changes to align sequences. However, the number of possible topologies increases exponentially as the number of sequences increases. Thus, this method is computationally extremely expensive. Later developed algorithms in this category make some restrictions to reduce the search space. One example is to assume that internal nodes can only derive states from known sequences, when the sequences are closely related and quite similar [69]. This method avoids the effect of the guide tree. Nevertheless, it is unable to identify the sub-optimal tree structures and alignments. The assumption that the states can only be derived from known sequences is biologically unrealistic.

Another approach is based on the Bayesian framework which maximizes the posterior probabilities of the tree and the alignment simultaneously. It assigns weights to possible alignments by their probabilities. The algorithm introduced by Redelings *et al.* [48], utilizes the MCMC method which searches through the sample spaces of parameters and tree topologies. The sampled random trees can safely guide the multiple alignment without introducing bias. It also applies the substitution model with site rates parameters in tree reconstructions.

However, this algorithm is impeded by its time requirement for the sampling of all parameters as well as the alignments and topologies. The experiment carried out by Redelings *et al.* [48] showed that processing the data set of five sequences took about one day in a PC with an AMD Athlon 1.7 GHz processor with one GB of RAM. Moreover, in modern biological analysis, the sequence data set is usually much larger than in this experiment. Nevertheless, this algorithm provides an inspiring initial work for producing the statistically reliable alignment and the tree structure at the same time. Further exploit is how to be more efficient in the search process.

## A.4 Aligning texts

In this section, we discuss a method for aligning text data. This method is similar to Clustal and Knees' method for aligning lyrics [34]. It first builds a guide tree, then aligns all texts from bottom to top. It achieves good results in aligning texts of the *Heinrichi* data set.

Before the progressive alignment method is applied to text data, we use letters to encode the words of the text. Depending on how large the text data is, different lengths of letters can be chosen to encode the texts. For example the two-letter codes allowing $26^2$ variables can be used for a relatively small text data. But for a larger data, four-letter code of $26^4$ variables may be more suitable. Similar words such as *womanhood* and *woman-hood*, are naturally considered as matching each other even though they are not exactly the same. Consider, for instance, the following three sentences:

This is an elephant
There is an elephant
There are elephants

The words *elephant* and *elephants* can be coded by the same three-letter code. Thus, they can be coded as:

aaa aac aae aaf
aab aac aae aaf
aab aad aaf

Thus, similar words are assigned with the same code in the coding step. After the coded texts are aligned, they are then decoded back to the original texts.

The progressive alignment procedure for the text alignment is similar to Clustal. Figure 15 gives an illustration of the steps in the program. First, a guide tree using the neighbor-joining method [54] is built. The distance between two texts is measured by the least information needed for changing between the two texts [49]. For instance, the amount of information needed for text $a$ to change to text $b$ is calculated by subtracting the bits needed for compressing $a$ alone from the bits of needed for compressing $a$ concatenated with $b$. The texts are aligned bottom-up until they reach the root. This strategy is to align more similar sequences first to avoid undetermined gaps or mismatches. However, errors are prone to be made in the first

steps because there are less already aligned sequences, thus less information than in the later steps to assure the right alignment. The re-alignment can be performed once all texts are aligned. Also the columns that contain only gaps should be removed from the final results. In the aligning process, two sets containing multiple already aligned texts need to be aligned. This can be done by modifying the global alignment algorithm, the Needleman-Wunsch algorithm, which is originally used for pairwise alignment of two sequences. We discuss the modified algorithm in next section. For the previous example, after the alignment, the set of sentences becomes:

| | | | |
|---|---|---|---|
| aaa | aac | aae | aaf |
| aab | aac | aae | aaf |
| aab | aad | - | aaf |

Then we can encode them back as:

| | | | |
|---|---|---|---|
| This | is | an | elephant |
| There | is | an | elephant |
| There | are | - | elephants |

This algorithm is similar to the methods developed by Spencer *et al.* [60] and Knees *et al.* [34]. In the method by Knees, they do not consider the similar words as matches in the alignment process. Any different word pair is treated as the mismatch even though they may be homologous in linearity. It leads to more gaps inserted in the alignment. To avoid the redundant gaps, the algorithm assigns less penalties for mismatches and more penalties for gaps. However, it may cause the distant words to be aligned because it does not take how similar two words are into account.

In the method developed by Spencer *et al.*, the similarities of the words are represented by 2-gram distances [66]. Each time two text sets are aligned, each pair of the words between the two sets needs to be compared to calculate the score based on their 2-gram distance. This may result in repeated comparisons of the words which are already known to be homologous in the previous steps. To speed up the process, the texts are divided into blocks by line or by paragraph because it is much easier to align small pieces than to align the whole texts. The small blocks are linked together at the end to form the whole alignment. For large data sets, for example the *Heinrichi* data set, our method is comparable with Spencer's method.

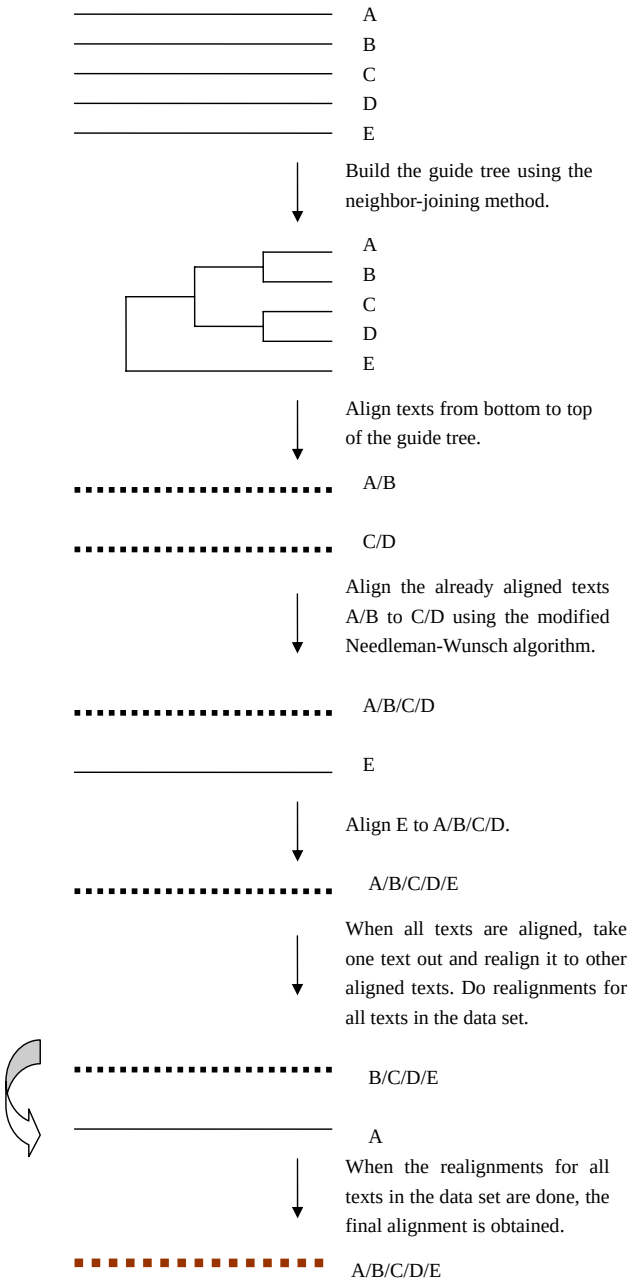



Figure 15: Schematic of the procedure for the multiple text alignment using the progressive method. The dashed line represents the already aligned texts.

## A.5 Modified Needleman-Wunsch algorithm

In this section, we first discuss the original Needleman-Wunsch algorithm [43] for aligning two strings before extend the algorithm for aligning two sets of strings. The Needleman-Wunsch algorithm is a global alignment algorithm based on the dynamic programming method. It builds a score matrix and searches the best path through this matrix. The size of the matrix for two strings of length $n$ and $m$ is $(n+1) \times (m+1)$, with the first row and column to enable gaps in the beginning of the strings. The element in left top conner can be initialized as zero, and the score matrix is gradually filled in from top to bottom and from left to right. Denote the two strings as $(a_1, ..., a_n)$ and $(b_1, ..., b_m)$, the element $s_{i,j}$ in the score matrix is the *maximum* value of the following three possible values:

- Define $\eta(a_i, b_j)$ as the match or mismatch score of $a_i$ and $b_j$. When $a_i$ and $b_j$ are aligned, $s_{i,j}$ is computed from the element on the upper left diagonal of it:

$$s_{i,j} = s_{i-1,j-1} + \eta(a_i, b_j). \tag{78}$$

- Define $\psi$ as the gap penalty. When $a_i$ is aligned with a gap, $s_{i,j}$ is computed from the element on the left of it:

$$s_{i,j} = s_{i,j-1} + \psi. \tag{79}$$

- When $b_i$ is aligned with a gap, $s_{i,j}$ is computed from the element on the top of it:

$$s_{i,j} = s_{i-1,j} + \psi. \tag{80}$$

Usually the match score is set to be more than zero, while the gap and mismatch scores are less than zero. After all the elements in the score matrix are computed, the optimal path for the alignment is found by tracing from the highest score element in the right-bottom corner to the first element in the left-top corner. The running time of the Needleman-Wunsch algorithm is $O(nm)$ and the amount of memory required is $O(nm)$.

To extend the algorithm to align two sets of already aligned texts, the score of aligning $a_i$ and $b_j$ can be modified as the average of the score of aligning all pairs of the words in the two sets. Assuming that we need to align two already aligned text *sets*: $(a^1, ..., a^n)$ and $(b^1, ..., b^m)$, where $a^i = (a^i_1, ..., a^i_{n'})$ and $b^j = (a^i_1, ..., a^i_{m'})$ for all

$1 \leq i \leq n$ and $1 \leq j \leq m$. The score for aligning the two sets can be computed as:

$$s_{i,j} = \frac{1}{n' \cdot m'} \sum_{u=1}^{n'} \sum_{v=1}^{m'} s(a_i^u, b_j^v), \tag{81}$$

where $s$ can be the match, mismatch or gap score. Therefore, the already aligned word sets which may contain gaps can also be incorporated in the alignment. The way for calculating the score when opening a gap is unchanged. After the modification, the Needleman-Wunsch algorithm can be applied in aligning two sets of aligned sequences in the same way as aligning two single sequences.